

Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering



Diploma Thesis

Efficient Application Programming in Python

Sunny AKA Shivam Dave

© 2020 CULS Prague

DIPLOMA THESIS ASSIGNMENT

B.Sc. Sunny Aka Shivam Dave

Systems Engineering and Informatics
Informatics

Thesis title

Efficient application programming in Python

Objectives of thesis

The goal of this thesis is to present the Python programming language as a new and rapidly growing application development platform which seems to be a strong competitor to Java/Javascript/PHP family to the near future.

Methodology

In the first part of this thesis, a deep analysis of various sources about Python will be performed. This first part will include; application development in Python, software metrics, debugging and profiling and possible comparison to their equivalents from Java/Javascript/PHP family. In the second and practical part of this thesis, a small application example in Python will be programmed. This small example should work with some information publicly accessible from the web and will use this Python application example.

The proposed extent of the thesis

80 – 100 pages

Keywords

Python; application programming; efficient programming; internet of things

Recommended information sources

BEAZLEY, David M. a Brian K. JONES. (2013) Python cookbook. 3rd. Sebastopol: O'Reilly. ISBN 9781449340377

DANJOU, Julien. (2019) Serious Python: black-belt advice on deployment, scalability, testing, and more. San Francisco, CA: No Starch Press, Inc. ISBN 1593278780;9781593278786.

EL-SHAROOD, W. (2019) Book review: Charles severance, Python for everybody. London, England. SAGE Publications, 277pp. ISBN 0036-8504

SLATKIN, Brett (2015) Effective Python: 59 specific ways to write better Python. Beaverton: Ringgold Inc. ISBN 2372-3424.

Expected date of thesis defense

2019/20 SS – FEM

The Diploma Thesis Supervisor

doc. Ing. Vojtěch Merunka, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 11. 3. 2020

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 11. 3. 2020

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 03. 04. 2020

Declaration

I declare that I have worked on my diploma thesis titled " Efficient Application Programming in Python " by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 04/04/2020

Sunny AKA Shivam Dave

Acknowledgement

I would like to begin by thanking my family for their constant help, support and love.

I would like to express my sincere thanks to my thesis supervisor doc. Ing. Vojtěch Merunka, Ph.D. at Czech University of Life Sciences Prague Faculty of Economics and Management for his great support and understanding in the completion of this thesis. His irreplaceable encouragement to lead me in the right ways has driven me to this success.

I would also like to thank my professors, my classmates and the academic staff of the Systems Engineering and Informatics department of the Czech University of Life Sciences, Prague, for their support during the last two years of my learning process.

I would like to finish by thanking my friends for their willingness to support me in every way I needed help. Without their encouragement and their important contributions, I could not have finished this thesis.

Efficient Application Programming in Python

Abstract

The purpose of this thesis is to present the Python programming language as a modern and rapidly growing application development framework that promises to be a strong competitor to the Java / Javascript / PHP family. A deep analysis of various sources about Python will be conducted in the first part of this thesis. This part will include: Python application development, performance metrics, debugging and profiling, and potential comparison with its Java / Javascript / PHP family equivalents. A small code example in Python will be programmed in the second and practical part of this thesis. This small code example will function with some web-accessible information, and use the Python Script.

Keywords: Python, application programming, efficient programming, RE(Regular Expression), internet of things, DS(Data Structures)

Efektivní programování aplikací v Pythonu

Abstrakt

Účelem této práce je představit programovací jazyk Python jako moderní a rychle rostoucí rámec pro vývoj aplikací, který slibuje, že bude silným konkurentem rodiny Java / Javascript / PHP. V první části této práce bude provedena hloubková analýza různých zdrojů o Pythonu. Tato část bude zahrnovat: vývoj aplikací Pythonu, metriky výkonu, ladění a profilování a potenciální srovnání s ekvivalenty rodiny Java / Javascript / PHP. Ve druhé a praktické části této práce bude naprogramován malý příklad kódu v Pythonu. Tento příklad malého kódu bude fungovat s některými informacemi přístupnými na webu a použije skript Python.

Klíčová slova: Python, programování aplikací, efektivní programování, RE (regulární výraz), internet věcí, DS (datové struktury)

Table of content

1	Introduction.....	8
2	Objectives and Methodology.....	9
2.1	Objectives.....	9
2.2	Methodology.....	9
3	Literature Review.....	10
3.1	Python History.....	10
3.1.1	Syntax and Semantics.....	11
3.1.2	Keywords and Identifiers.....	12
3.1.3	Data Structures in Python.....	13
3.1.4	Mapping Operators to Functions.....	16
3.2	Manipulation of string and built-in methods.....	20
3.2.1	Working with file & handling in python.....	22
3.2.2	Python Lists and Tuples.....	27
3.2.3	Set and Dictionaries.....	32
3.3	Internet Socket.....	36
3.4	Regular Expression(regex).....	41
3.5	Profiling.....	44
4	Practical Part.....	46
4.1	Open-Source Intelligence(OSINT).....	46
4.2	Tools and Technologies.....	48
4.2.1	Installing Python.....	48
4.2.2	Installing ATOM(text-editor) for Python.....	49
4.2.3	Installing PyCharm(IDE) for Python.....	50
4.2.4	Installing JAVA and IntelliJ IDEA(IDE).....	52
4.3	Code Implementation.....	53
4.3.1	Implementation in a Python.....	54
4.3.2	Implementation in JAVA.....	56
4.4	Accomplishment.....	61
4.4.1	Result from String[a-z,A-Z].....	62
4.4.2	Result from Numbers[0-9].....	64
4.4.3	Result in Google's search engine.....	65
4.5	Profiling Comparison.....	69
5	Results and Discussion.....	74
5.1	Discussion.....	75
5.1.1	TIOBE Index.....	75

6 Conclusion.....	76
7 References	77

List of Figures

Figure.1 Python standard type hierarchy (standard-type).....	12
Figure.2 Implementation of Walrus Operator.....	19
Figure.3 While loop Implementation with Walrus Operator.....	19
Figure.4 Conditional Statement Implementation with walrus operator.....	20
Figure.5 Indexing in lists and string (Indexing-lists).....	27
Figure.6 HTTP request/response and data retrieval.....	39
Figure.7 urllib library to retrieve data	40
Figure.8 urllib, treat like a file and make a histogram logic with words.....	41
Figure.9 OSINT Reconnaissance.....	47
Figure.10 Executable files-python installer.....	48
Figure.11 Python Interpreter-CMD.....	49
Figure.12 Executable files-Atom installer.....	49
Figure.13 Atom Text Editor.....	50
Figure.14 Executable files- PyCharm-community edition.....	51
Figure.15 PyCharm Community Edition 2019.3.1.....	51
Figure.16 JAVA 8 installation.....	52
Figure.17 Executable files-Intellij IDEA community edition.....	53
Figure.18 Intellij IDEA- Community Edition-2019.3.1.....	53
Figure.19 browser.py.....	54
Figure.20 Regular Expression in Google.py.....	55
Figure.21 def search() method.....	56
Figure.22 Maven Configuration pom.xml file.....	57
Figure.23 Browser.java.....	58
Figure.24 Google.java_1.....	59
Figure.25 Google.java_2.....	60
Figure.26 Google.java_3.....	61
Figure.27 Build result of browser.py(string contain[a-z,A-Z]).....	62
Figure.28 Build result of Browser.java(string contain[a-z,A-Z]).....	63
Figure.29 Build result of browser.py(string contain[0-9]).....	64
Figure.30 Build result of Browser.java(string contain[0-9]).....	64
Figure.31 Result Collage_1 String[a-z,A-Z].....	65
Figure.32 Result Collage_2 String[a-z,A-Z].....	66
Figure.33 Result Collage_3 String[a-z,A-Z].....	67
Figure.34 Result_4 String[0-9].....	68
Figure.35 Result_5 String [0-9].....	68
Figure.36 Profiling in Python.....	69
Figure.37 Profiling Result in CMD.....	70

Figure.38 pstats.Stats Module(SortKey).....	71
Figure.39 Jprofiler IDE integration.....	71
Figure.40 Jprofiler-Methods Cumulative time.....	72
Figure.41 Jprofiler-package impact on memory.....	73
Figure.42 Jprofiler-Cumulated methods outgoing Graph.....	73
Figure.43 TIOBE Programming Community Index.....	75

List of tables

Table.1 Reserved Words/Keywords.....	13
Table.2. Data Types.....	14
Table.3. Built-in Datatypes.....	16
Table.4. Operator Symbols (PythonDOC).....	18
Table.5 Built-in file handling libraries (Built-in-lib).....	26
Table.6 Common TCP Ports (TCP-portNum).....	37
Table.7 Metacharacters in Regular expression.....	42
Table.8 Method/Attribute in Regular Expressions.....	43
Table.9 IDE-System Requirements.....	51

1 Introduction

In this modern age of innovation and technological transformations, the role of programming languages evolved in the last 5 decades. Computers are designed for one reason- to do tasks for People, yet human must speak their language to explain what should be done. A sequence of stored instructions is a little piece of human's intelligence in the computer.

Users see computers as a series of instruments-word processor, table, chart, to - do list, and so on. Programmers learn the "forms" of the system and the programming language. Programmers have some software to create new applications, often programmers write software for a users and sometimes programmers write little "software" to automate a task for themselves. Programmers are creating a piece of creative art-particularly when they do a good job regarding user experience.

Python is evolving more than last two decades aim to create a computer programming language for everyone which is easy and readable language, as powerful as major open source rivals. Python is as comprehensible as plain English language for daily tasks so python community nurtures and community members can contribute to the development code.

Python supports multiple programming paradigm such as Procedural programming, Imperative Programming, and object-oriented programming. Python is a General-Purpose Programming language which helps to create application for multiple domains like- desktop app, console app, web app, mobile app, and also used for modern domain such as IoT, Artificial Intelligence, Data Science, Cybersecurity.

In last 5 years, Internet generated such a gigantic amount of data that never been generated before in a digital world. Python has a largest repository compare to major other open source languages named as Python Package Index (PyPI) which is free and available to developer/programmer. Profound analysis on python(3.8.1)'s data structure, technique and new features discussed in this writing with illustration.

2 Objectives and Methodology

2.1 Objectives

The aim of this thesis is to introduce the Python programming language as a modern and exponentially growing environment for application development which in the close future expects to be a strong competitor to the Java / Javascript / PHP family.

2.2 Methodology

A systematic review of the various references concerning Python will be carried out in the first part of this study. This first section would include: Python program creation, performance analytics, debugging, profiling, and potential contrast with its Java / Javascript / PHP family equivalents. A brief code sample in Python will be programmed in the second and functional portion of this study. This sample code will work with any publicly accessible information from the internet via Google's search engine and use Python framework for implementation.

3 Literature Review

3.1 Python History

Python was founded as a successor of a language called ABC by Guido van Rossum at the Stichting Mathematisch Centrum in the Netherlands in the early 1990's. Guido remains the primary developer of Python, though he includes many other contributions. Guido began his research on Python in 1995 at the National Research Initiatives Corporation in Reston, Virginia where he launched several versions of the language.

Guido Van Rossum submitted a funding request to DARPA(Defence Advanced Research Projects Agency) in 1999 called "Computer Programming for Everyone," in which he further described his goals for Python: An simple and understandable language as strong as major Open Source rivals, so that anyone can contribute to their development code, which is as understandable as plain English for everyday tasks, enabling fast development times. Guido and the core development team at Python moved to BeOpen.com in May 2000 to form the BeOpen PythonLabs team. The PythonLabs team moved on to Digital Creations (now Zope Corporation) in October of the same year.

The Python Software Foundation (PSF) was founded in 2001, a non-profit organization created specifically for controlling intellectual property relating to Python. PSF endorsing member is Zope Corporation. All versions of Python are Open Source. Historically most, but not all, versions of Python were also GPL compatible. As of 3 December 2008, Python 3.0 was released. It was a major language update, which is not entirely backward compatible. Many of its major features have been backporting to version series Python 2.6.x and 2.7.x. Python 3 updates include the 2to3 utility that automates the conversion (at least partially) of Python 2 code into Python 3.

The end-of-life deadline of Python 2.7 was initially set at 2015 and then delayed until 2020 out of concern that a large body of existing code could not be easily forward-loaded to Python 3. Python 3.8 was released on October 14th, 2019.

3.1.1 Syntax and Semantics

Python is designed to be an easy to read language. The formatting is visually uncluttered, and often uses English keywords while punctuation is used by other languages. Unlike many other languages, the curly brackets are not used to delimit lines, and semicolons are optional after sentences. It has fewer exceptions to syntactics and special cases than other languages. Instead of curly brackets or keywords, Python uses white space indentation to delimit the segments. The increase in indentation occurs after some statements; a decrease in the indentation of the end of the current segment. Therefore, the visual structure of the program accurately represents the semantic structure of the programme. This aspect is sometimes called off-side law, which is shared by some other languages, but indentation has no semantic sense in most languages.

Python employs duck typing; A programming style that does not look at the type of an object to decide whether it has the right interface; rather, it simply calls or uses the method or attribute ("If it looks like a duck and quacks like a duck it must be a duck."). Through emphasizing interfaces instead of specific types, well-designed code improves their versatility through allowing polymorphic replacement. Type constraints are not checked at the time of compilation; instead, operations on an object may fail, meaning the given object is not of an appropriate type. Although being dynamically typed, Python is strongly typed, preventing non-defined operations (for example, adding a number to a string) rather than implicitly trying to make sense of them.

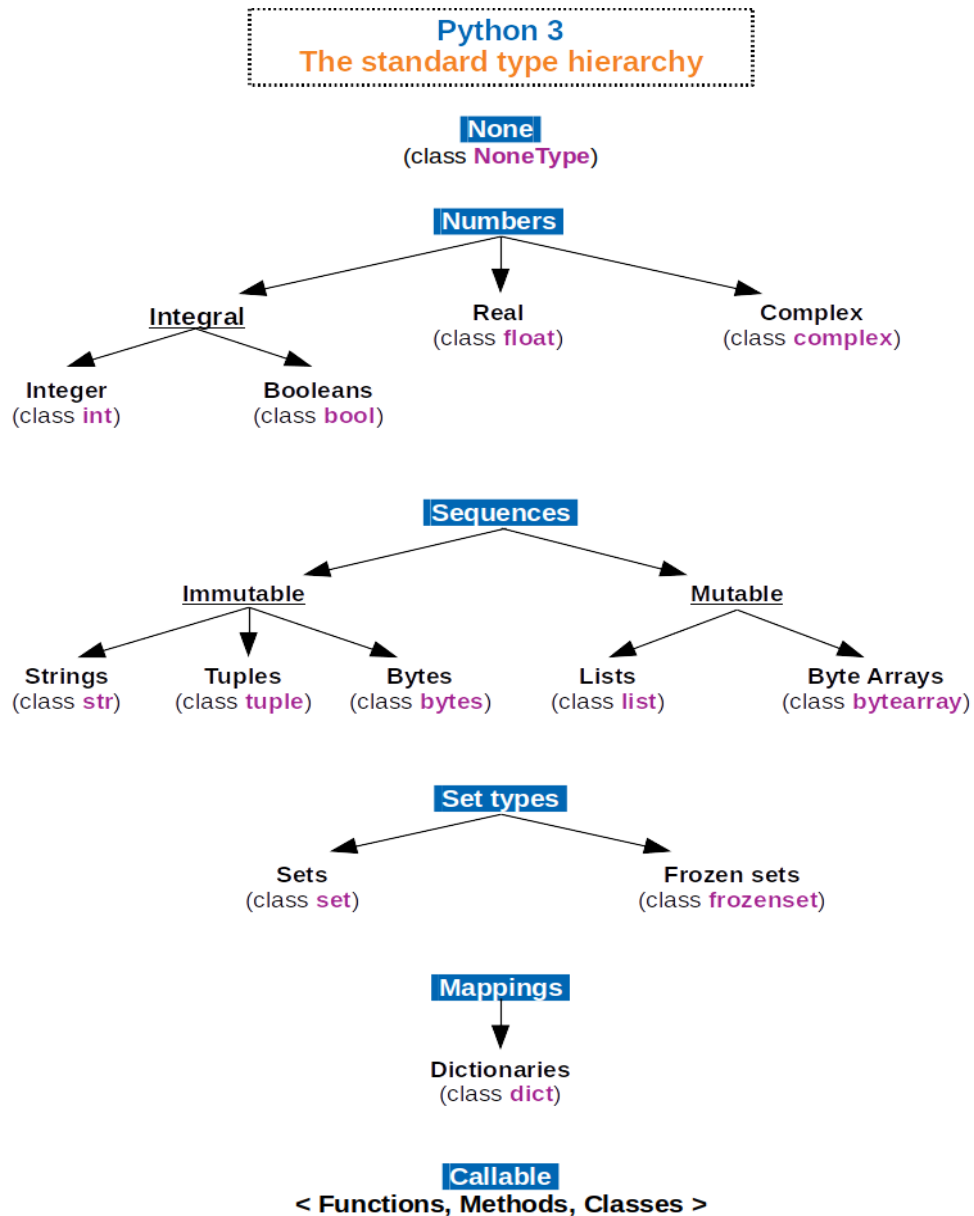


Figure.1 Python standard type hierarchy (**standard-type**)

3.1.2 Keywords and Identifiers

We cannot use a keyword to describe the syntax and structure of the Python language as variable names, function names, or any other identifier. In Python, keywords are case sensitive. All the keywords are in lowercase except True, False, and None, and they must be written as they are. The list below is given of all the keywords. **(Keywords)**

True	False	None	and	as
assert	async	await	break	class
continue	def	del	elif	else
except	finally	for	from	global
if	import	in	is	lambda
nonlocal	not	or	pass	raise
return	try	while	with	yield

Table.1 Reserved Words/Keywords

An identifier is a name for entities such as class, functions, variables etc. It assists in separating one entity from another. Identifiers may be a combination of lower-case letters (a to z) or uppercase letters (A to Z) or digits (0 to 9) or underscore `_`. These names as (myClass, var_1 and print_this_hello) are all legitimate examples. An identifier must not begin with a digit. (1var) is invalid but (var1) is fine. Unable to use keywords as identifiers. Unique signs such as (!, @, #, %, \$, etc.) can't be used in the identifier. Do not use single characters such as l, O as 1 and 0 can be confused.

3.1.3 Data Structures in Python

Python offers some embedded data types as well, especially (dict, frozenset, list, set and tuple). Unicode strings are stored in the `<str>` class and binary data is stored in the bytes and bytearray classes. Python provides a wide range of specific data types, such as dates and times, fixed arrays, heap queues, double-ended queues and enumerations.

Data for different types can be stored in variables and different types can do different things. Python has built-in data types in these categories, by default as mentioned below:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict

Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

Table.2. Data Types

For Python every value has a datatype. Since everything for Python programming is an object, data types are in fact classes, and variables are instance (object) of these classes. There are several types of data in Python. The following are some of the main forms.

Datatype	Description	Mutability	Example
str	A character string: sequence of Unicode codepoints	immutable	<pre>>>> x = "Hello People" >>> print(x) Hello People >>> print(type(x)) <class 'str'></pre>
int	Integer of unlimited magnitude	immutable	<pre>>>> x=21 >>> print(x) 21 >>> print(type(x)) <class 'int'></pre>
complex	Complex number with real and imaginary parts	immutable	<pre>>>> x = 2+8j >>> print(x) (2+8j) >>> print(type(x)) <class 'complex'></pre>
list	List, can contain mixed types	mutable	<pre>>>> x = ['beer', 'wine', 'whiskey'] >>> print(x) ['beer', 'wine', 'whiskey'] >>> print(type(x)) <class 'list'></pre>
dict	Associative array (or dictionary) of key and	mutable	<pre>>>> x = {"name" : "Shivam", "age" : 25}</pre>

	value pairs; can contain mixed types (keys and values), keys must be a hashable type		<pre>>>> print(x) {'name': 'Shivam', 'age': 25} >>> print(type(x)) <class 'dict'></pre>
tuple	Can contain mixed types	immutable	<pre>>>> x = ('beer', 'wine', 'whiskey') >>> print(x) ('beer', 'wine', 'whiskey') >>> print(type(x)) <class 'tuple'></pre>
range	A Sequence of numbers commonly used for looping specific number of times in for loops	immutable	<pre>>>> x = range(5) >>> print(x) range(0, 5) >>> print(list(x)) [0, 1, 2, 3, 4] >>> print(type(x)) <class 'range'></pre>
Bool	Boolean value, True or False	immutable	<pre>>>> x = True >>> print(type(x)) <class 'bool'></pre>
set	Unordered set, contains no duplicates; can contain mixed types, if hashable	mutable	<pre>>>> x = {'beer', 'wine', 'whiskey'} >>> print(x) {'wine', 'whiskey', 'beer'} >>> print(type(x)) <class 'set'></pre>
frozenset	Unordered set, contains no duplicates; can contain mixed types, if hashable	immutable	<pre>>>> x = frozenset({'beer', 'wine', 'whiskey'}) >>> print(x) frozenset({'wine', 'whiskey', 'beer'})</pre>

			<pre>>>> print(type(x)) <class 'frozenset'></pre>
float	<p>Double precision floating point number. The precision is machine dependent but in practice is 64 bits.</p>	immutable	<pre>>>> x = 20.6668 >>> print(x) 20.6668 >>> print(type(x)) <class 'float'></pre>
bytes	Sequence of bytes	immutable	<pre>>>> x = b'Shivam' >>> print(x) b'Shivam' >>> print(type(x)) <class 'bytes'></pre>
bytearray	Sequence of bytes	mutable	<pre>>>> x = bytearray(4) >>> print(x) bytearray(b'\x00\x00\x00\x00') >>> print(type(x)) <class 'bytearray'></pre>
memoryview	exposes the C level buffer interface as a Python object	immutable	<pre>>>> x = memoryview(bytes(3)) >>> print(x) <memory at 0x0000021B17128280> >>> print(type(x)) <class 'memoryview'></pre>

Table.3. Built-in Datatypes

we can get the data type of any object by using the type() function. In python, some datatypes like (None, NotImplemented, ellipsis) are not directly accessible by name.

3.1.4 Mapping Operators to Functions

In Python, operators are special symbols which perform arithmetic or logical computation. The values which are applied to the operator are called operands. Table

shows how abstract operations correspond to operator symbols in the Python syntax and the functions in the operator module.

Operation	Syntax	Function
Addition	<code>a + b</code>	<code>add(a, b)</code>
Concatenation	<code>seq1 + seq2</code>	<code>concat(seq1, seq2)</code>
Containment Test	<code>obj in seq</code>	<code>contains(seq, obj)</code>
Division	<code>a / b</code>	<code>truediv(a, b)</code>
Division	<code>a // b</code>	<code>floordiv(a, b)</code>
Bitwise And	<code>a & b</code>	<code>and_(a, b)</code>
Bitwise Exclusive Or	<code>a ^ b</code>	<code>xor(a, b)</code>
Bitwise Inversion	<code>~ a</code>	<code>invert(a)</code>
Bitwise Or	<code>a b</code>	<code>or_(a, b)</code>
Exponentiation	<code>a ** b</code>	<code>pow(a, b)</code>
Identity	<code>a is b</code>	<code>is_(a, b)</code>
Identity	<code>a is not b</code>	<code>is_not(a, b)</code>
Indexed Assignment	<code>obj[k] = v</code>	<code>setitem(obj, k, v)</code>
Indexed Deletion	<code>del obj[k]</code>	<code>delitem(obj, k)</code>
Indexing	<code>obj[k]</code>	<code>getitem(obj, k)</code>
Left Shift	<code>a << b</code>	<code>lshift(a, b)</code>
Modulo	<code>a % b</code>	<code>mod(a, b)</code>
Multiplication	<code>a * b</code>	<code>mul(a, b)</code>
Matrix Multiplication	<code>a @ b</code>	<code>matmul(a, b)</code>

Operation	Syntax	Function
Negation (Arithmetic)	- a	neg(a)
Negation (Logical)	not a	not_(a)
Positive	+ a	pos(a)
Right Shift	a >> b	rshift(a, b)
Slice Assignment	seq[i:j] = values	setitem(seq, slice(i, j), values)
Slice Deletion	del seq[i:j]	delitem(seq, slice(i, j))
Slicing	seq[i:j]	getitem(seq, slice(i, j))
String Formatting	s % obj	mod(s, obj)
Subtraction	a - b	sub(a, b)
Truth Test	obj	truth(obj)
Ordering	a < b	lt(a, b)
Ordering	a <= b	le(a, b)
Equality	a == b	eq(a, b)
Difference	a != b	ne(a, b)
Ordering	a >= b	ge(a, b)
Ordering	a > b	gt(a, b)

Table.4. Operator Symbols (PythonDOC)

In Python 3.8, There is a new syntax (:=) that assigns values as part of a larger expression to the variables. Due to its resemblance to the eyes and tusks of a walrus it is affectionately known as "the walrus operator." The Assignment expressions allow a value to be assigned to a variable, even a variable that doesn't exist yet, in the context of expression rather than as a stand-alone statement.

```
1 #Demo 1
2
3 print("Demo 1")
4
5 # Without Walrus
6 x = 10
7 not_walrus = x < 15
8 print(not_walrus)
9
10 # with Walrus
11 X = 10
12 print(Walrus := x < 15)
```

Run: walrus ×

C:\Users\shiva\PycharmProjects\osint\venv\Scripts\python.exe C:/Users/shiva/PycharmProjects/osint/pkg/walrus.py

Demo 1
True
True

Figure.2 Implementation of Walrus Operator

```
14 # Demo 2
15 print("Demo 2")
16 # Without Walrus
17 number = []
18 num = input('Enter a number:')
19 while num.isdigit():
20     number.append(int(num))
21     num = input('Enter a number:')
22 print(number)
23
24 # with Walrus ## with walrus operator we write less line of code and achieve same output.
25 number = []
26 while (num := input('Enter a number:')).isdigit():
27     number.append(int(num))
28 print(number)
```

Run: walrus ×

Demo 2
Enter a number:5
Enter a number:4
Enter a number:6
Enter a number:5
[5, 4, 6]
Enter a number:96
Enter a number:45
Enter a number:6
Enter a number:7
[96, 45, 6]

Figure.3 While loop Implementation with Walrus Operator

```
29
30 # Demo 3
31 print("Demo 3")
32 # Without Walrus
33 var = 5
34 if var == 5:
35     ans = input("Enter your Answer:")
36     if ans != "":
37         print(ans)
38 # with Walrus
39 var = 5
40 if var == 5 and (ans := input("Enter your Answer:")) != "":
41     print(ans)
42
```

Run: walrus x

```
Enter your Answer:
5
Enter your Answer:
True
```

Figure.4 Conditional Statement Implementation with walrus operator

3.2 Manipulation of string and built-in methods

Python can also manipulate strings alongside numbers, which can be represented in several ways. They can be enclosed with the same outcome in single quotes (' ... ') or double quotes ("...") 'sample' is the same as "sample", with the print() function we can display string literal. As with many other common programming languages, Python strings are byte arrays containing Unicode characters. Python has no sort of data character; however, a single character is simply a string with a length of 1. Strings may be indexed with index 0 for the first letter. Square brackets [] may be used for accessing string elements.

The output string is enclosed in quotes in the interactive interpreter and special characters get escaped with backslashes (\). If we don't want to read characters prefaced by (\) as special characters, we can use raw strings by inserting a 'r' before the first quote: >>> print(r'C:\users\shiva\name') will produce output: C:\users\shiva\name. using a raw string 'r' python interpreter won't confuse with \n as a new line. Strings literal can span

several lines. Another approach is to use triple quotes: `"""..."""` or `'...'`. The `in` keyword can also be used to check to see if one string is `in` another string.

With the `+` operator strings can be concatenated and repeated with `*` operator, python can only concatenated strings with a same datatype, other datatypes give us the traceback. We can concatenate different datatypes using type conversion. Concatenation only works with two literals though, not with variables or expressions. With `+` operator we can concatenate variables or a variable and a literal. Concatenation literally glued strings together without whitespaces if we want to add whitespace, we need to add it manually as shown in below example.

```
>>> x = 'Welcome to'
>>> y = x + 'Prague'
>>> print(y)
Welcome toPrague    # no whitespace between str1(x) and str2(y)
>>> y = x + ' ' + 'Prague'
>>> print(y)
Welcome to Prague   # whitespace added between str1(x) and str2(y)
```

We can also look at any continuous section of a string using a colon (`:`) operator is known as string slicing. Slicing helps you to obtain a substring when indexing is used to obtain individual characters. While performing slicing on a string start of index element is always included and end of index element is always excluded, this ensures that `s[:i] + s[i:]` is always the same as `s[:]`. One way to remember how slices operate is to think of the indices as pointing between characters, with the first character's left edge numbered 0. Then the right edge of a string of `n` characters has index `n`. However, when used for slicing, indexes of slices out of range are treated gracefully. Python strings are unalterable, they are immutable. Therefore, an error occurs in assigning an indexed location in the string.

```
>>> x = 'shivam'
>>> x[0:2]
'sh'
>>> x[2:5]
'iva'
>>> x[2:]
'shivam'
>>> x[0:200]
```


'shivam'

One of the real advantages of python3 is all strings are Unicode, which means they can represent wide range of character sets. Strings implement all the common sequence operations, along with the additional 40 built-in methods well described in official python documentation and summary mentioned below. (**String-Methods**)

```
>>> x = 'Welcome to'
>>> y = x + ' ' + 'Prague'      # concatenation string with whitespace
>>> print(y)
Welcome to Prague
>>> type(y)
<class 'str'>
>>> dir(y)      #this line display all built-in methods supported by class 'str'
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattr__', '__getitem__', '__getnewargs__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__',
'__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize',
'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format',
'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier',
'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower',
'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit',
rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper',
'zfill']
```

3.2.1 Working with file & handling in python

Python is an excellent data-processing tool. Any software we write is likely to involve reading, writing, or manipulating the data. For this reason, knowing how to handle different file formats which store different data types is especially useful. Python is extremely accommodating and can handle a range of different file formats with relative

ease, including following but not limited to: txt(Text file), CSV(Comma-separated values file), HTML(HyperText Markup Language File), JSON(JavaScript Object Notation File).

The key feature for interacting with Python files is the `open()` function. The `open()` returns a “file handle” - a variable used to perform operations on the file. handle is something that's sort of a porthole between program and file that's sitting on the disk. And we can open it, we can read from it, we can write to it if we want. And then we can close the handle and the handle goes away, but the handle is like our connection.

The function `open()` requires two arguments; `open(filename, mode)`. The first argument is a string with the filename on it. The second argument is another string which contains a few characters that describe how the file will be used. Mode can be 'r' when the file is read only, 'w' when it is written only (an existing file with the same name will be deleted), and 'a' opens the file to be appended; any data written to the file will be automatically added to the end. 'x'- mode create the required file and returns an error if the file already exists the file 'r+' mode opens to both read and write. The mode argument is optional; if it is removed 'r' read only will be presumed. we can also decide whether the file should be dealt with as binary using 'b' or text mode using 't', by default.

```
>>> f = open('sample.txt')
>>> print(f)
<_io.TextIOWrapper name='sample.txt' mode='r' encoding='UTF-8'>
>>>f = open("sample.txt")      # equivalent to 'r' or 'rt'
>>>f = open("sample.txt",'w')  # write in text mode
>>>f = open("img1.png",'r+b')  # read and write in binary mode
```

Usually files are accessed in text mode, meaning read and write strings from and to the file that are encoded in a specific encoding. 'b' attached to the mode opens the file in binary mode: the data is now read and written in byte object format. This mode should be used on all files not containing text. Using the ‘with’ keyword is a good practice when working with file objects in python. The benefit is that, even if an exception is posed at

some point, the file is properly closed after its suite finishes. It is also much easier to use with than to write equivalent try-finally blocks. **(Read&write-I/O)**

```
>>> with open('sample.txt') as f:
...     read_data = f.read()
>>> f.closed    #using this verify that file has been automatically closed
True
```

If we don't use the 'with' keyword, need to call `f.close()` to close the file and release any system resources that it utilizes by it immediately. If you do not specifically close a file, the garbage collector from Python will eventually destroy the object and close the open file for you, but the file will remain open for some time. Another danger is that different versions of Python can clean up this at different times. After a file object is closed, attempts to use the file object will automatically fail either by a statement or by calling `f.close()`. **(Read&write-I/O)**

Methods of file objects we will suppose a file object named 'f' has been created before. Read the contents of a file, call `f.read(size)`, which reads a certain amount of data and returns it as an object string (in text mode) or bytes (in binary mode). Size is an additional numerical argument. If size is ignored or negative, the entire content of the file will be read and returned; if the file is twice the size of our machine's memory, it is our concern. Otherwise characters of a maximum size (in text mode) or bytes of size (in binary mode) are read and returned. If the file end is reached, `f.read()` returns an empty string ('').

```
>>> f.read()
'Hello Pythonist, welcome to the programming world\n'
>>> f.read()
''
#returns an empty string
```

`f.readline()` reads a single line from the file; a newline character (`\n`) is left at the end of the string and is skipped only on the last line of the file if the file does not end in a

newline. This makes the return value indisputable; if `f.readline()` returns an empty string, the end of the file is reached, while `'\n'` represents a blank line, a string that includes only one newline.

```
>>> f.readline()
' Hello Pythonist, welcome to the programming world\n '
>>> f.readline()
'let's learn this awesome language called python.\n'
>>> f.readline()
''          #returns an empty string
```

we can loop over the file object to read lines from a file. This is efficient in memory, fast and leads to simple code.

```
>>> for line in f:
...     print(line, end=")
...
Hello Pythonist, welcome to the programming world.
'let's learn this awesome language called python. (Read&write-I/O)
```

`f.write(string)` writes the string contents to the file and returns the number of characters written.

```
>>> f.write("This is a test\n")
15
```

Using the `seek()` method, we can adjust our current file cursor (location). Likewise, the `tell()` method returns our current position in number of bytes.

```
>>> f.tell() # get the current file position
12
>>> f.seek(5) # bring file cursor to initial position
```

we need to import the OS module to delete a file and run its `os.remove()` function. Use the method `os.rmdir('foldername/directoryname')` to remove an entire folder/directory. Check whether file exists, then delete it shown below.

```
>>>import os
>>>if os.path.exists("sample.txt"):
>>>    os.remove("sample.txt")
>>>else:
>>>    print("No file found")
```

There are also built-in libraries out there which we can use to help. There's just a lot more out there. In addition, more third-party tools are available on PyPI(<https://pypi.org/>). Some of the more popular ones are: `pyPDF2`(PDF toolkit), `xlwings`(read and write Excel files), `pillow`(image reading and manipulation).

Built-in Libraries	File format type
wave	read and write WAV files (audio)
aifc	read and write AIFF and AIFC files (audio)
sunau	read and write Sun AU files
tarfile	read and write tar archive files
zipfile	work with ZIP archives
configparser	easily create and parse configuration files
xml.etree.ElementTree	create or read XML based files
msilib	read and write Microsoft Installer files
plistlib	generate and parse Mac OS X .plist files

Table.5 Built-in file handling libraries (**Built-in-lib**)

3.2.2 Python Lists and Tuples

Python's most flexible, useful data types are lists and tuples. Virtually any nontrivial Python program consist them. Python knows many types of compound data which are used to group certain values together. The most flexible is the list which can be written in between square brackets as a list of comma-separated values (items). Lists can contain items of various types but typically all items have the same type. List is a collection which is ordered and mutable. Tuple is a collection which is ordered and immutable. Both allows duplicate members.

Like strings and all other built-in sequence types, lists can be indexed and sliced. All slice operations return a new list which contains the elements requested. Which means a shallow copy of the list returns. Lists facilitate operations such as concatenation(+), replication(*) and len(), min(), max() function. it allows assignment operation to slice[:], and this can also alter the size of the list or clear it completely. Lists can be nested to arbitrary depth. Virtually everything about indexing in strings works likewise for lists. A negative list index for instance counts from the end of the list.

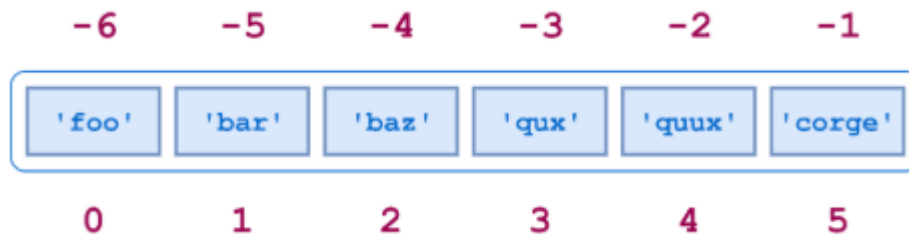


Figure.5 Indexing in lists and string (**Indexing-lists**)

```
>>> x = ['1', '2', '3', 'A', 'B', 'c', 'd', '3.14']
>>> x[3:6] = ['4', '5', '6']          # replace items
>>> print(x)
['1', '2', '3', '4', '5', '6', 'd', '3.14']
>>> x[3:6] = []          # remove item
>>> print(x)
['1', '2', '3', 'd', '3.14']
>>> x[:] = []          # clear the list by replacing all the elements with an empty list
>>> print(x)
```

```

[]
>>> print(x[::-1])    #reverse the list items
[3.14, 'd', 'c', 'B', 'A', '3', '2', '1']

```

The sort of list data type has a few additional methods. Lists implement all the common sequence operations, along with the additional built-in methods well described in official python documentation and summary mentioned below. **(list-methods)**

```

>>> print(type(x))
<class 'list'>
>>> dir(x)    #this line display all built-in methods supported by class 'list'
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',
 '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__gt__',
 '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__',
 '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
 '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
 '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',
 'remove', 'reverse', 'sort']

```

The methods of list make it very easy to use a list as stack, where the last added element is the first retrieved element "last-in, first-out". Use append() method to add an element to the top of that stack. Using pop() method without a specific index to retrieve an object from the top of the Stack. A list can also be used as a queue, where the first element inserted is the first item that has been retrieved "first-in, first-out"; however, lists are not effective for this reason. Although appends and pops from the end of the list are quick, making appends or pops from the beginning of a list is slow because all the other elements need to be moved by one. Using collections.deque which was built to have fast appends and pops from both ends to implement a queue. **(class-collection-deque)**

List comprehensions provide a concise way of producing lists. Popular applications are to create new lists where each element is the product of certain operations applied to each member of a different sequence or iterable, or to create a sequence of those elements that satisfy a certain condition. An element in a list can be an object of any type, It requires

a different list. A list may contain sublists that may, in turn, contain sublists themselves, and so on to an arbitrary depth. We can create a list of squares using list comprehension like never before. (**nestedlist-comp**)

```
>>> lst = [x**2 for x in range(10)]    # more concise and redable
>>> lst
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Or

```
>>> lst1 = list(map(lambda x: x**2, range(10)))    #using lambda function
>>> lst1
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Every arbitrary expression, including another list comprehension, can be the initial expression in a list comprehension. Consider the following example of an applied 4x3 matrix as a set of 4 columns of length 3.

```
>>> m = [
... [99,98,97],
... [89,88,87],
... [99,98,97],
... [69,68,67],
... ]
>>> m
[[99, 98, 97], [89, 88, 87], [99, 98, 97], [69, 68, 67]]
>>> [[row [i] for row in m] for i in range(3)]    # transpose row and columns
[[99, 89, 99, 69], [98, 88, 98, 68], [97, 87, 97, 67]]
```

The nested list comprehension is evaluated in the following sense, so this below example is equivalent to the above example.

```
>>> transposed1 = []
>>> for i in range(3):
...     transposed1.append([row[i] for row in m]) # (nestedlist-comp)
```


...

```
>>> transposed1
```

```
[[99, 89, 99, 69], [98, 88, 98, 68], [97, 87, 97, 67]]
```

str.Split() method divides a string into parts, and produces a string list and we can parse strings. We consider these terms as words. Through all the words we can access a specific word or loop. If we don't specify a delimiter, multiple spaces are treated as one delimiter, we may decide which delimiter(,;,@,.,etc.) character to use when splitting.

Lists and strings have many common characteristics, including indexing and slicing operations. They are two examples of types of sequence data. Python is an evolving language it is possible to add other categories of sequence data type. There is another standard type of data type, too: the **tuple**. A **tuple** is composed of several values separated by commas. Tuples are immutable sequences which are usually used to store heterogeneous data collections, Tuples are also used in cases where an immutable sequence of homogeneous data such as allowing for storage in a set or dict instance is needed. **(sequence-types)**

In all respects tuples are equivalent to lists, except for the following properties: 1.) Instead of square brackets ([]), tuples are identified by putting elements in parentheses(). 2.) Tuples are immutable.

```
>>> a = ('ab','bc','cd','ef')
```

```
>>> print(a)
```

```
('ab', 'bc', 'cd', 'ef')
```

```
>>> print(type(a))
```

```
<class 'tuple'>
```

```
>>> dir(a)      #this line display all built-in methods supported by class 'tuple'
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',  
'__format__', '__ge__', '__getattr__', '__getitem__', '__getnewargs__', '__gt__',  
'__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',  
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__',  
'__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

In a certain scenario we should use tuple instead of list, 1.) When processing a tuple, program execution is faster than the identical list. If the list or tuple is small, this probably won't be noticeable. 2.) Sometimes we do not want to alter data. If the stored values are intended to remain constant over the lifespan of the program, we use tuple rather than a list, it guards against unintended alteration.

```
>>> t = (1,2,3,4), a    #nested tuples
>>> t
((1, 2, 3, 4), ('ab', 'bc', 'cd', 'ef'))
```

The tuple building of 0 or 1 objects is a unique problem. Empty tuples are constructed by an empty parenthetic pair(). Tuple of one entity is formed by include a trailing comma(,) just before the closing parenthesis. If comma is not added python confused with operator precedence in expression.

```
>>> t1 = ('shivam','vivek','aman','roger')    # tuple packing
>>> t1[0]
'shivam'
>>> (p1,p2,p3,p4) = t1    # tuple unpacking
>>> p2
'vivek'
```

When unpacking a tuple, the number of variables on the left must match the number of values in the tuple. Packing and unpacking a combined in to one statement to make a compound assignment.

```
>>> (i1,i2,i3,i4) = ('10dce019','3.14','pi','7004') # compound assignment
>>> i2
'3.14'
```

we often have two variables values that we need to swap while programming. One of the values in a temporary variable should be saved in most programming languages during the swap. In Python, the swap can be done with a single tuple assignment.

```
>>> fname = 'Sunny AKA Shivam'
>>> lname = 'Dave'
>>> fname, lname
('Sunny AKA Shivam', 'Dave')
>>> fname, lname = lname, fname # magical statement, swap value without 3rd var
>>> fname, lname
('Dave', 'Sunny AKA Shivam')
```

While tuples that appear similar to lists, they are often used for different situations and purposes. Tuples are immutable but they can contain mutable objects like tuple consist of multiple lists and usually contain a heterogeneous sequence of accessible elements by unpacking or indexing. Lists are mutable, and their elements are typically homogeneous and accessible through the list iteration.

3.2.3 Set and Dictionaries

Python also includes a set as a datatype. A **set** is an unordered, non-indexed, non-duplicate collection. python use curly braces{ } or the set() function to construct sets. Common uses include checking membership, eliminating duplicates from a sequence, and computing mathematical operations such as intersection, union, difference and symmetrical difference.

By referring to an index, we cannot access objects in a set, because sets are unordered, and the items have no index sets and do not support indexing, slicing, or other sequence-like behaviour. But by using the 'for' keyword, we can loop through the set items, or by using 'in' keyword we ask if a given value is in a set. we can't change the items once a set is formed but we can add new items. len() function used to determine the how many items set has, we use remove() or discard() method to remove an item from a set. The only difference between the two is that it remains unchanged when using discard() if the item

does not exist in the set. But in such case remove() would cause an error. Method pop() also remove a last item from the set, sets are unordered so using pop() method we don't know which item that gets removed. Method clear(), removes all items from set while 'del' keyword will delete the set totally.

Python has various ways to join two or more sets, union() method returns a new set including all items from both sets, update() method inserts all the items from one set into another set. The set() constructor may also be used to make a set.

```
>>> s = {'samsung','google','apple','microsoft'}      # define set
>>> print(type(s))
<class 'set'>      # (set-methods)
>>> dir(s)      #this line display all built-in methods supported by class 'set'
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',
 '__format__', '__ge__', '__getattr__', '__gt__', '__hash__', '__iand__', '__init__',
 '__init_subclass__', '__ior__', '__isub__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__',
 '__ne__', '__new__', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__',
 '__ror__', '__rsub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__',
 '__subclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_update',
 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset',
 'pop', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
 'update']
```

Frozenset is a new class with the characteristics of a set, but once allocated, its elements cannot be altered. Although tuples are lists which are immutable, frozensets are sets which are immutable. Mutable sets are unhashable, so they cannot be used as dictionary keys. Frozensets, on the other hand, may be hashable and used as dictionary keys. We use the frozenset() function to construct frozensets.

```
>>> fs = frozenset(['s20ultra','pixel4xl','iphone11pro','surfacepro'])
>>> print(type(fs))      # define frozenset
<class 'frozenset'>      # (frozenset-methods)
```

```

>>> dir(fs)    #this line display all built-in methods supported by class 'frozenset'
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',
 '__format__', '__ge__', '__getattr__', '__gt__', '__hash__', '__init__',
 '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__',
 '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor__',
 '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__xor__', 'copy',
 'difference', 'intersection', 'isdisjoint', 'issubset', 'issuperset', 'symmetric_difference',
 'union']

```

Similarly, to list comprehensions, set comprehensions are also supported. frozenset Being immutable it does not have method that add or remove items.

Dictionaries are python's most powerful data collection. Dictionaries allow us to do fast database-like operations in python. Python dictionary is an unordered collection of items. While other types of compound data only have value as an element, a dictionary has one key : value pair and written with curly{ } brackets. A colon (:) separates each key from its associated value. Dictionaries are sometimes known as "properties or map or HashMap-JAVA " or "associative arrays-Perl/PHP," or "property bag-C#/.NET" in other languages. Dictionaries are indexed by keys, which can be any immutable type, unlike sequence data types indexed by a range of numbers; strings and numbers can always be keys. Tuples may be used as keys if they only contain strings, numbers or tuples; if a tuple includes any mutable entity directly or indirectly, it cannot be used as a key. Lists cannot be used as keys, as lists can be modified in order using index assignments, slice assignments, or methods such as append() and extend().

The easiest way to think of a dictionary as a collection of key : value pairs, allowing the keys to be unique within one dictionary. A couple of braces generate an empty dictionary:{}. Placing a comma-separated list of key : value pairs within braces adds initial key : value pairs to the dictionary; this is also how dictionaries are written on output.

```

>>> d = {'shivam': 25, 'vivek': 23, 'nikoleta': 25, 'aman': 29}
>>> print(type(d))

```

```

<class 'dict'>
>>> dir(d)      #this line display all built-in methods supported by class 'dict'
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__gt__', '__hash__',
 '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__setattr__', '__setitem__',
 '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items',
'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']      # (dict-mapping-type)
>>> d1 = dict(shivam=25, vivek=23, nikoleta=25, aman=29)
>>> d2 = dict(zip(['shivam', 'vivek', 'nikoleta', 'aman'], [25,23,25,29]))
>>> d3 = dict([('shivam', 25), ('vivek', 23), ('nikoleta', 25), ('aman',29)])
>>> d == d1 == d2 == d3      # various ways to define dictionary
True

```

Dictionaries and lists share the following properties: both are mutable and dynamic, they can grow and shrink as needed, both can be nested; dictionary can contain another list or dictionary, and vice versa. The main difference between the dictionaries and the lists is how they get to the elements: List elements are accessed by indexing through their position in the list. Access to dictionary elements by keys.

Dictionaries have several additional methods specific to their structure. Methods that return lists like items, keys, and values, are not guaranteed to do so in any particular order, but may be in a consistent order if no modifications are made to the dictionary in between the calls. The get() method is often preferable to index notation because it does not raise an error when the requested key is not found; instead it returns None by default, or a default value that is passed as a second argument. We loop through the key-value pairs in a dictionary using **two** iteration variables. Each iteration, the first variable is the key and the second variable is the corresponding value for the key. We can use get() and provide a default value of zero when the key is not yet in the dictionary - and then just add one as mentioned in below example.

```
>>> counts = dict()
```

```

>>> names = ['shivam', 'vivek', 'aman', 'vivek', 'vivek', 'nikoleta']
>>> for name in names :
...     counts[name] = counts.get(name, 0) + 1           # default value 0
...
>>> print(counts)
{'shivam': 1, 'vivek': 3, 'aman': 1, 'nikoleta': 1} # histogram of names

```

3.3 Internet Socket

“In computer networking, an Internet socket or network socket is an endpoint of a bidirectional inter-process communication flow across an Internet Protocol-based computer network, such as the Internet.” **(Charles Russell Severance, September 9, 2013)** TCP port numbers/port is an process specific or application specific software communication endpoint. It allows the co-existence of many networked applications on the same server.

Internet Assigned Numbers Authority(IANA) is responsible for the global coordination of of the IP addressing, Domain Name System(DNS) root and other internet protocol resources. This includes the registration for known Internet services with frequently used port numbers. Port Numbers categorised in to three ranges: well-known ports(0 to 1023), registered ports(1024 to 49151), ephemeral/private ports(49152 to 65535). IANA maintains the official list of well-known and registered ports. Web or other uniform resource locators (URLs) also display port numbers. Default is port 80 for HTTP and port 443 for HTTPS. Sometimes we see the port number in URL if the web server is runnig on “Non-standard” port.

Port Number	Description
20	File Transfer protocol(FTP) Data Transfer
21	File Transfer protocol(FTP) Command Control
22	Secure Shell(SSH), secure login
23	Telnet remote login service, unencrypted text message
25	Simple Mail transfer Protocol(SMTP), E-mail routing
53	Domain Name System(DNS) Service

67,68	Dynamic Host Configuration Protocol(DHCP)
80	HyperText Transfer Protocol(HTTP) used in World Wide Web(WWW)
109,110	Post office Protocol(POP3), Mail Retrieval
119	Network News Transfer Protocol(NNTP)
123	Network Time Protocol(NTP)
143,220,993	Internet Message Access Protocol(IMAP), management of e-mail
161	Simple Network Management Protocol(SNMP)
194	Internet Relay Chat(IRC)
443	Secure HTTPS

Table.6 Common TCP Ports (**TCP-portNum**)

Python is fantastic Let's say we want to use millions of lines of code in our computer's connection and network and transport layer, as well as the whole internet and some server on the other side, and the data and we want to talk to them? Python takes three lines. Python has built-in support for TCP sockets.

```
>>>import socket (Charles Russell Severance, September 9, 2013)
>>>mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>>mysock.connect( ('data.pr4e.org', 80) )
```

Socket usage began with ARPANET in 1971 and later became an API in the operating system Berkeley Software Distribution (BSD), released in 1983 called Berkeley sockets. When the Internet started with the World Wide Web in the 1990's, network programming did likewise. Web servers and browsers were not the only applications that gained from newly linked networks and were using sockets. Widespread use came with client-server systems of all types and sizes.

Even though the underlying protocols used by the socket API have changed over the years, and we've seen new ones, the low-level API remains the same today. Client-server applications are the most common type of socket applications, where one side serves as the

server and awaits client connections. The Python socket module provides the Berkeley sockets API with an interface.

```
# Create an INET, STREAMing socket
>>>mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#now connect to the web server on port 80 -the normal http port
>>>mysock.connect( ('data.pr4e.org', 80) )
```

For the AF_INET for IPv4 address set, a pair (host, port) is used, where host is a string representing either a hostname in the Internet domain notation such as 'data.pr4e.org' or an IPv4 address such as '192.241.136.170' and port is an integer from 1-65535. Two special types are accepted for IPv4 addresses instead of a host address: '' represents INADDR_ANY which is used for binding to all interfaces, and the string '<broadcast >' represents INADDR_BROADCAST. This activity is not IPv6 compliant, so if we wish to support IPv6 with our Python programs, we might want to avoid these.

A four-tuple (host, port, flowinfo, scopeid) is used for the AF_INET6 family of IPv6 addresses. Flowinfo and scopeid can be removed for the socket module methods just for backward compatibility. Note, however, scopeid omission may cause problems when manipulating scoped IPv6 addresses. A primary socket API Functions, Objects and methods are well described in official documentation of docs.python.org and some of them discussed below. (**Socket-methods**)

socket.socket() creates a socket object that supports the context manager type, if it uses it in a 'with' statement there is no need to invoke s.close() method. The arguments passed to socket() define family of addresses and type of socket. AF_INET is the IPv4 family of web addresses. SOCK_STREAM is the type of TCP socket, the protocol that will be used to carry our messages over the network. Bind() is used to connect a socket to a specific network interface and port number.

Listen() allows server to accept() connections. It creates a "listening" socket. Listen() has the parameter Backlog. This determines the amount of unaccepted connections that

will be enabled by the program before rejecting new connections. It is optional, beginning in Python 3.5. If not stated a default value for the backlog is selected. Accept() blocks an incoming connection and waits. When a client connects, a new socket object representing the connection is returned, and a tuple containing the client's address. The tuple will contain for IPv4 connections (host, port), or for IPv6 (host, port, flowinfo, scopeid).

```

socket.py      socket2.py      regexp.py
1
2 import socket
3
4 mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 mysock.connect(('data.pr4e.org', 80))
6 cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()
7 mysock.send(cmd)
8
9 while True:
10     data = mysock.recv(512)
11     if len(data) < 1:
12         break
13     print(data.decode(),end=' ')
14
15 mysock.close()
16
Command Prompt
02/02/2020  12:12 AM                466 xml2.py
              131 File(s)                7,367,592 bytes
              8 Dir(s)              70,058,610,688 bytes free

C:\Users\shiva\OneDrive\Desktop\py4e\code3\code3>socket1.py
HTTP/1.1 200 OK
Date: Wed, 11 Mar 2020 21:22:17 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sat, 13 May 2017 11:22:22 GMT
ETag: "a7-54f6609245537"
Accept-Ranges: bytes
Content-Length: 167
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: close
Content-Type: text/plain

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

C:\Users\shiva\OneDrive\Desktop\py4e\code3\code3>

```

Figure.6 HTTP request/response and data retrieval (Charles Russell Severance, September 9, 2013)

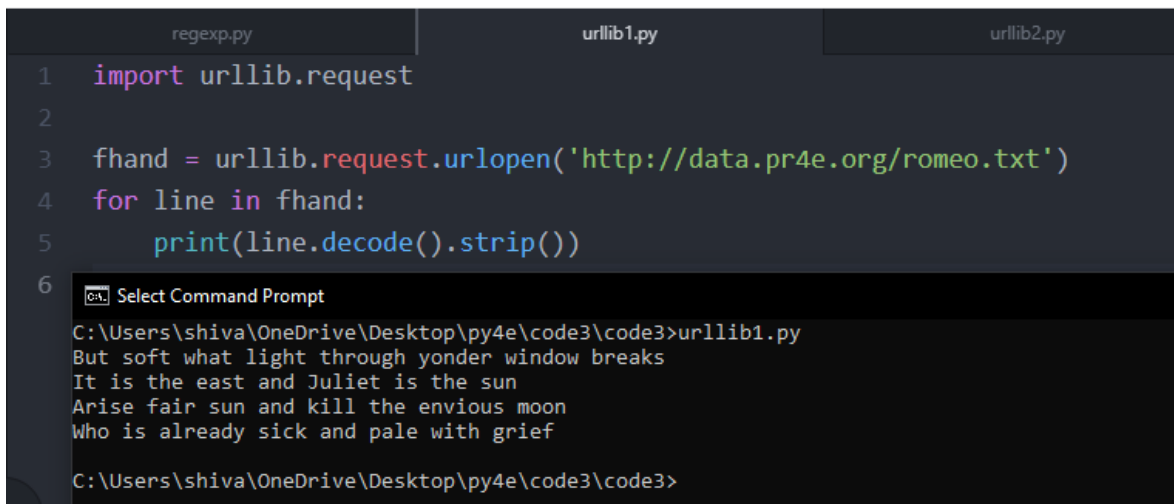
```

# this line makes the doorway between your system and server, it's like a Porthole
>>>mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# connect() basically extends out of your computer, it fails if server doesn't exist
>>>mysock.connect( ('data.pr4e.org', 80) ) # finds the server and connect to port 80
# GET request to get the content of the page
>>>cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()
# call methods on socket object send(), able to send
>>>mysock.send(cmd)

```

With the while loop and receive() method print the data from the romeo.txt, we specify parameter in receive() method, here it is 512 so it is going to receive up to 512 characters. Strings inside python are Unicode, encode() converts from Unicode to UTF-8. decode() converts UTF-8 to Unicode and print it in the form of python strings so we can apply all string methods on the data that we received from the web. In python it takes only 10 lines of code to perform HTTP request/response cycle and retrieve data from the web. In output cmd window HTTP header and HTTP body separated by blank line.

Since HTTP is so popular, python has a library called urllib that works for us and makes web pages look like a file. we use urllib to view a web page just like a file and simply indicate which web page we want to access, and urllib manages all the HTTP protocol and header. By using urllib we can reduce the lines of code and achieve same output with three lines of code. The comparable code and output from the web using urllib to read the romeo.txt file is as follows:



```
regexp.py      urllib1.py      urllib2.py
1  import urllib.request
2
3  fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
4  for line in fhand:
5      print(line.decode().strip())
6
  Select Command Prompt
C:\Users\shiva\OneDrive\Desktop\py4e\code3\code3>urllib1.py
But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

C:\Users\shiva\OneDrive\Desktop\py4e\code3\code3>
```

Figure.7 urllib library to retrieve data (Charles Russell Severance, September 9, 2013)

```

regexp.py      urlwords.py
1 import urllib.request, urllib.parse, urllib.error
2
3 fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
4
5 counts = dict()
6 for line in fhand:
7     words = line.decode().split()
8     for word in words:
9         counts[word] = counts.get(word, 0) + 1
10 print(counts)

```

```

C:\Users\shiva\OneDrive\Desktop\py4e\code3\code3>urlwords.py
{'But': 1, 'soft': 1, 'what': 1, 'light': 1, 'through': 1, 'yonder': 1, 'window': 1, 'breaks': 1, 'It': 1, 'is': 3, 'the': 3, 'east': 1, 'and': 3, 'Juliet': 1, 'sun': 2, 'Arise': 1, 'fair': 1, 'kill': 1, 'envious': 1, 'moon': 1, 'Who': 1, 'already': 1, 'sick': 1, 'pale': 1, 'with': 1, 'grief': 1}
C:\Users\shiva\OneDrive\Desktop\py4e\code3\code3>

```

Figure.8 urllib, treat like a file and make a histogram logic with words

We can also browse, download and parse internet data such as XML, HTML, JSON, etc. with Python. We can also use Python to work directly with that data.

3.4 Regular Expression(regex)

In programming, a regular expression, also known as "regex" or "regexp," provides a succinct and versatile means for matching text strings, such as unique letters, words, or character patterns. A regular expression is written in a formal language and can be interpreted by a processor for regular expression. Regular Expression is very powerful and quite cryptic to understand. It is a language of “marker characters”- programming with characters. Regular expression or Regex for short is a way to search pattern in a big text of data. regex is also used for data validation like secure password, correct email format but it's usage can be extended for penetration testing.

In python, regular expression patterns are compiled into a series of bytecodes which are then executed by a matching engine written in C. For advanced usage, close attention can need to be paid to how the engine should execute a given RE and write the RE in some way to generate bytecode that runs faster. Below mentioned is the complete list of metacharacters.

. ^ \$ * + ? { } [] \ | ()

Symbols	Description
.	Matches any character except new line
\d	Digit[0-9]
\D	Not a digit[^0-9]
\w	Word character[a-z,A-Z,0-9, _]
\W	Not a Word character[^a-z,A-Z,0-9, _]
\s	Whitespace(space, tab, newline)[\t\n\r\f\v]
\S	Not a whitespace(space, tab, newline) [^ \t\n\r\f\v]
\b	Word boundary
\B	Not a word boundary
^	Beginning of a string
\$	End of a string
[]	Matches character in a bracket
[^]	Matches character not in a bracket
	Either or
()	Group- tell where to start and stop what string to extract
<u>Quantifiers</u>	
Symbols	Description
*	0 or more(greedy searching)
+	1 or more
?	0 or one(non-greedy searching)
{5}	Curly braces for exact number
{3,5}	Range of numbers(Minimum, Maximum)

Table.7 Metacharacters in Regular expression

Before using regular expression in Python, must import the library/module using “import re”. The re module provides an interface to the regular expression engine. Regular expressions are compiled into pattern objects which have methods for specific operations, such as pattern matches or string substitutions. re.compile() also accepts an optional flag argument, which is used to allow various special features and variations in syntax. The RE

is passed as a string into `re.compile()`. REs are treated as strings since regular expressions are not part of the core language of Python, and no special syntax has been developed to express them. Much like the `socket` or `zlib` modules, the RE module is simply a C extension module included with Python because there are applications that don't need REs at all. Therefore, the language specifications need not be hindered by including them.

Once an object representing a compiled regular expression, pattern objects have several methods and attributes. Only the most significant ones below mentioned.

Method	Description
<code>match()</code>	Determine if the RE matches at string initialization.
<code>search()</code>	Search through a string, checking for any position where this RE matches
<code>findall()</code>	Find all substrings and return them as a list where the RE matches.
<code>finditer()</code>	Find and return all substrings where the RE matches as an iterator.
<u>Attribute work on a match object</u>	
Attribute	Description
<code>group()</code>	Return the string matched by the RE
<code>start()</code>	Return the starting position of the match
<code>end()</code>	Return the ending position of the match
<code>span()</code>	Return a tuple containing(start, end) positions of the match
<u>Modifying strings</u>	
Method	Description
<code>split()</code>	Split the string into a list, split it wherever the RE matches
<code>sub()</code>	Find all substrings where the RE matches, and replace them with a different string
<code>subn()</code>	Does the same thing as <code>sub()</code> , but returns the new string and the number of replacements

Table.8 Method/Attribute in Regular Expressions

`match()` and `search()` return `NONE`, if no match can be found. Instance is returned containing information about the match if match object is found. If a special regular

expression character to just behave normal character prefix it with '/'. Complete list of re module content well described in python documentation. **(re-module-content)** Jeffrey Friedl's Mastering Regular Expressions, written by O'Reilly, is almost definitely the most comprehensive book on regular expressions. Unfortunately, it concentrates solely on the flavors of regular expressions of Perl and Java, and does not include any Python content at all, so it won't be useful as a guide for Python programming.

3.5 Profiling

CProfile and profile provide the Python programs with a deterministic profiling. A profile is a collection of statistics that explains how often different parts of the system were performed and for how long. Those statistics can be compiled through the pstats module into reports. The basic Python library features two separate implementations of the same profiling framework; For most applications, CProfile is recommended; it's a C extension with a fair overhead, making it ideal for profiling long running programs. Profile, a pure Python module whose interface imitates CProfile , but adds substantial overhead to the profiled programs. If attempting to extend the profiler in some way, this module could make the task easier.

CProfile allows user rapidly perform profiling on an existing application. The first line shows that it has tracked calls. Of such calls some is primitive, which implies the call was not triggered by recursion. The next line: Ordered by; standard name, shows that the output was sorted using the text string in the far-right column. The headings in the columns shall include: **ncalls**, for the number of calls. **tottime**, for the total time spent in the given function (and excluding time made in calls to sub-functions). **percall**, is the quotient of tottime divide by ncalls. **cumtime**, is the cumulative time in this and all subfunctions(from invocation till exist). **percall**, is the quotient of cumtime divided by primitive calls. **filename:lineno(function)**. Provides the respective data of each function. **(profiling)**

If the first column contains two numbers (e.g. 3/1), it implies that the function has recurred. The second value is the primitive number of calls, and the former is the total

number of calls. Remember that those two values are the same when the feature does not reoccur, and only the single figure is written.

The `pstats.Stats` class reads the results of profiles from a file and formats them in various ways. The `Stats` class of the `pstats` module has a range of methods to modify and print the data stored in a file of profile results well described in official document (**profiling**).

```
p.sort_stats(SortKey.CUMULATIVE).print_stats(10)
```

Above mentioned line Sorts the profile into a function by accumulated time, and then prints out the ten most appropriate lines. The aim of deterministic profiling is to represent the fact that all function call, function return, and exception events are monitored, and accurate timings are made for the intervals between these events (during which time the user's code is executing).

In Python, when an interpreter is active during execution, the existence of instrumented code is not necessary for deterministic profiling to be carried out. Python immediately returns a key for each case (optional call back). Moreover, Python's interpreted design tends to add too much overhead to execution, that deterministic profiling tends to add only low overhead processing in traditional applications. The result of deterministic profiling is not so complex but offers expensive and intense information on the execution of a Python programme.

Stats of call counts can be used to find bugs in code and to identify potential inline expansion points (high call counts). Internal time statistics should be used to define "tight loops," which can be configured with caution. Cumulative time statistics can be used in algorithm selection to detect high level errors. Note that the uncommon treatment of cumulative time in this profiler allows results to be compared directly to iterative implementations for recursive implementations of algorithms.

4 Practical Part

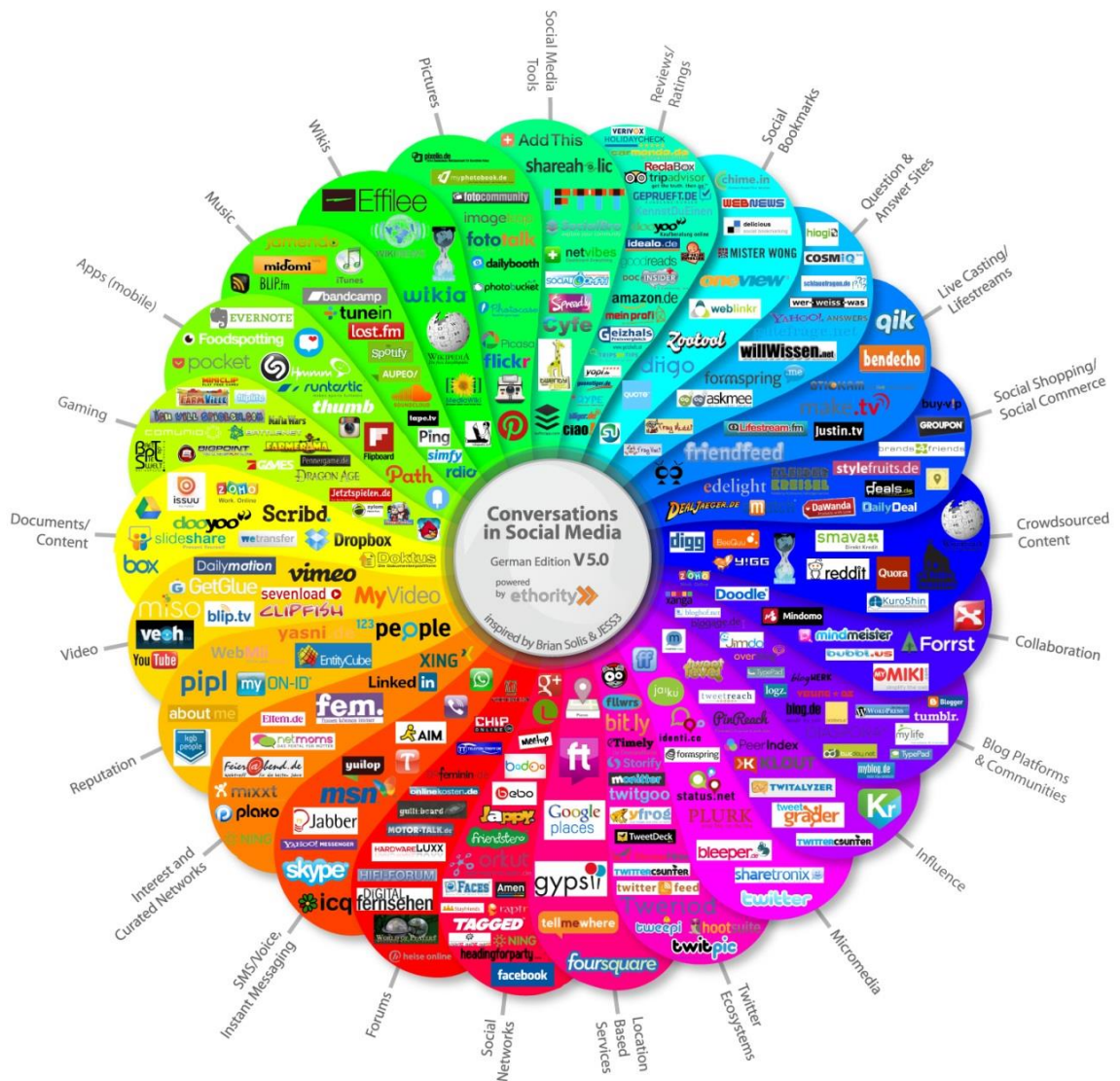
4.1 Open-Source Intelligence(OSINT)

Open source intelligence is generated from the data and resources accessible to the public at large. It isn't limited to what can be found using Google, but significant component is so-called "surface web". Open-source intelligence (OSINT) is data collected from publicly available sources to be used in an intelligence context. In the intelligence community, the term "open" refers to overt, publicly available sources as opposed to covert or clandestine sources.(OSINT) It is not related to open-source software or collective intelligence.

As useful as data from open source may be, overloading of information is a major concern. Most open source intelligence resources and techniques are designed to help security practitioners (or threat actors) concentrate their attention on particular areas of interest. Open source information has a dark side: Threat actors can still find (and use) something that can be identified by security professionals.

Crucially, the knowledge about open source is not limited to searching only the main search engines. Web pages and other sites that can be accessed using Google are huge sources of open source information but far from being the only outlets. For example, the big search engines cannot locate a huge proportion of the internet (over 99 percent, according to former Google CEO Eric Schmidt). This so-called "deep web" is a collection of websites, databases, files and more that for a number of reasons, including the existence of login pages or paywalls can't be indexed by Google, Bing, Yahoo or any other search engine to think about. Despite this, much of the deep web material can be called open source, as it is freely accessible to the public.

There's also plenty of freely available online information that can be collected using web resources other than conventional search engines. Resources such as Shodan and Censys can be used to locate IP addresses, networks, open ports, webcams, printers, and almost anything else connected to the Internet.



Conversations in Social Media - Version 5.0 - 09.2012 by ethority | <http://social-media-prisma.ethority.de> | <http://www.twitter.com/ethority> | Contact us for updates: prisma@ethority.de

Figure.9 OSINT Reconnaissance (Reconnaissance)

Here in the practical part, deploying a python script with Regular Expression(RE) that can use google as a search engine and trying to grab a possible URL link connection with a alpha-numeric string input provided in a python console.

4.2 Tools and Technologies

4.2.1 Installing Python

Install Python on Windows, Linux/Unix, MAC OS X, and other platforms like AIX, IBM i-series, iOS, Solaris VMS, HP-UX, etc. all version for the different platform available on <https://www.python.org/downloads/>. For this practical using Windows 10 as a host operating system and system configuration below mentioned.

Processor: Intel(R) Core(TM) i5-4210 CPU @1.70GHz

Disk Drive: Intel SSD-SC2BW120A4 120.00GB

Installed memory(RAM): 8.00GB

System Type: 64-bit Operating System, x64-based Processor

External GPU, Pen and Touch: Not applicable

Installing python on Windows 10 it required executable file(.exe) mentioned below to setup python environment on the host system.

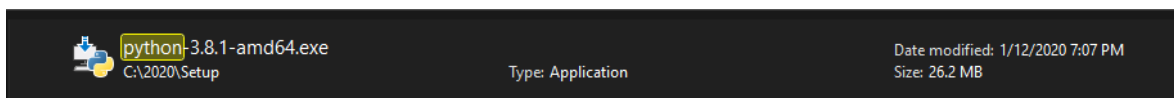
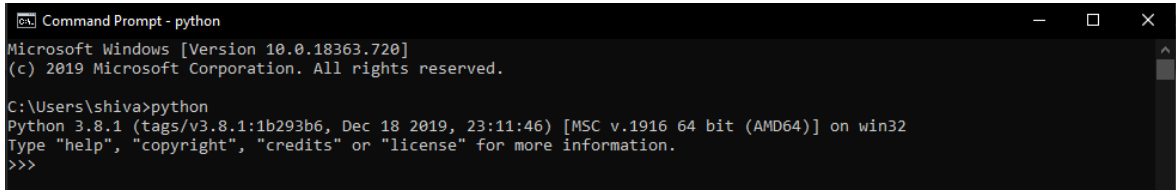


Figure.10 Executable files-python installer

Execute the .exe file for Python installer and it will pop up the window with installation step guide and it's a nice thing to add python 3.8 to PATH so we aware about the path where it is going to install on a host system. By clicking on a next python installer download all necessary component like interpreter, development libraries, executable, standard libraries, bootstrap and add to PATH. And its quick installation process through executable file.

After the installation process completed, Host System can launch python application from start menu or writing python in cmd prompt will launch the python interpreter within cmd. It will display the current version of the python and release date.



```
Command Prompt - python
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shiva>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure.11 Python Interpreter-CMD

4.2.2 Installing ATOM(text-editor) for Python

Atom is a free and open-source text and source code editor. Available for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is Covering all process of text editing and source code editing; it is a desktop application built using web technologies. Atom is a tool to customize to do anything but also use productively without ever touching a configuration file.

Atom Setup time should not be too long, it is quick and easy. Atom has Responsive design; Atom is ubiquitous text editor so user can use it from anywhere and on any platform. Installation Atom on windows all version for the different platform available on <https://atom.io/>. Installing Atom on Windows it required executable file(.exe) mentioned below to setup Atom on the host system.

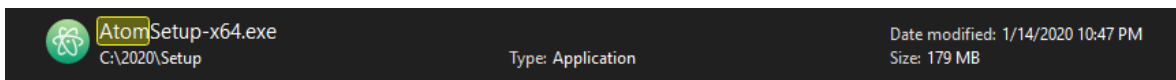


Figure.12 Executable files-Atom installer

To avoid indention errors in a python most of the text editors can turn tabs in to spaces; for that user must need to make sure to enable this feature from settings and preferences. Despite Atom automatically uses spaces for files with “.py“ extension.

Execuete the .exe file for Atom installation and follow the process once it is finished it will launch the below application.

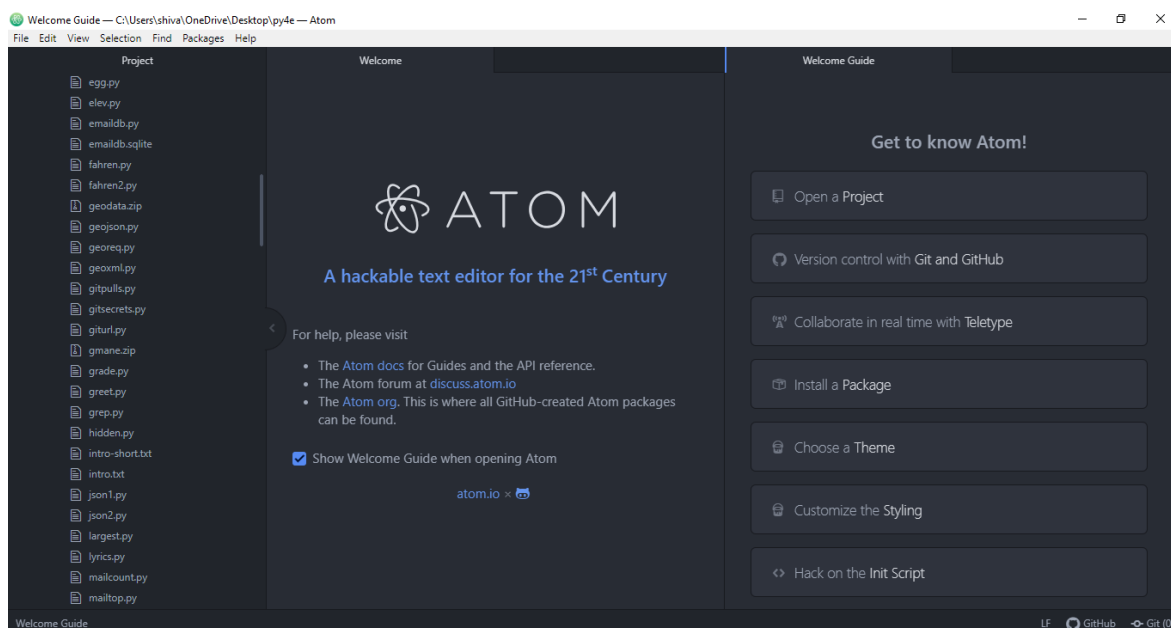


Figure.13 Atom Text Editor

4.2.3 Installing PyCharm(IDE) for Python

PyCharm is a cross-platform IDE, offering clear Windows, macOS, and Linux operating systems experience. JetBrains offers PyCharm in three versions: Professional, Community, and Edu. The versions of Community and Edu are open-source projects and are free but have less features. PyCharm Edu offers courses and helps learn how to program with Python. The Professional Version is private and offers an excellent selection of tools and features.

Requirement	Minimum	Recommended
RAM	4 GB of free RAM	8 Gb of Total
Disk Space	2.5GB and another 1GB for caches	SSD Drive with at least 5GB of free space
Monitor resolution	1024x768	1920X1080
Operating system	Officially released 64-bit versions of the following: - Microsoft Windows 7 SP1 or later	Latest 64-bit version of Windows, macOS, or Linux (for example, Debian, Ubuntu, or RHEL)

	<ul style="list-style-type: none"> - macOS 10.11 or later - nay Linux distribution that supports Gnome, KDE or Unity DE 	
--	---	--

Table.9 IDE-System Requirements

To run PyCharm, no need to install Java because JetBrains Runtime is bundled with IDE (based on JRE 11). For standalone installation of PyCharm, it requires the installer available on <https://www.jetbrains.com/pycharm/download/#section=windows> we can download the .exe for community version which is open source below mentioned.

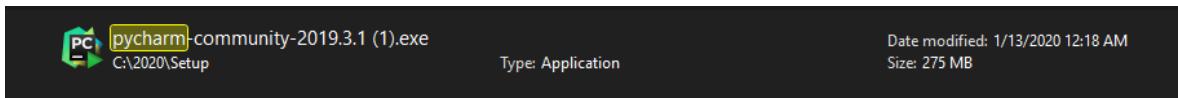


Figure.14 Executable files- PyCharm-community edition

Once installation is finished, host system can launch the PyCharm IDE and it asked for theme preferences, packages, and setup the project interpreter with built-in python console and terminal.

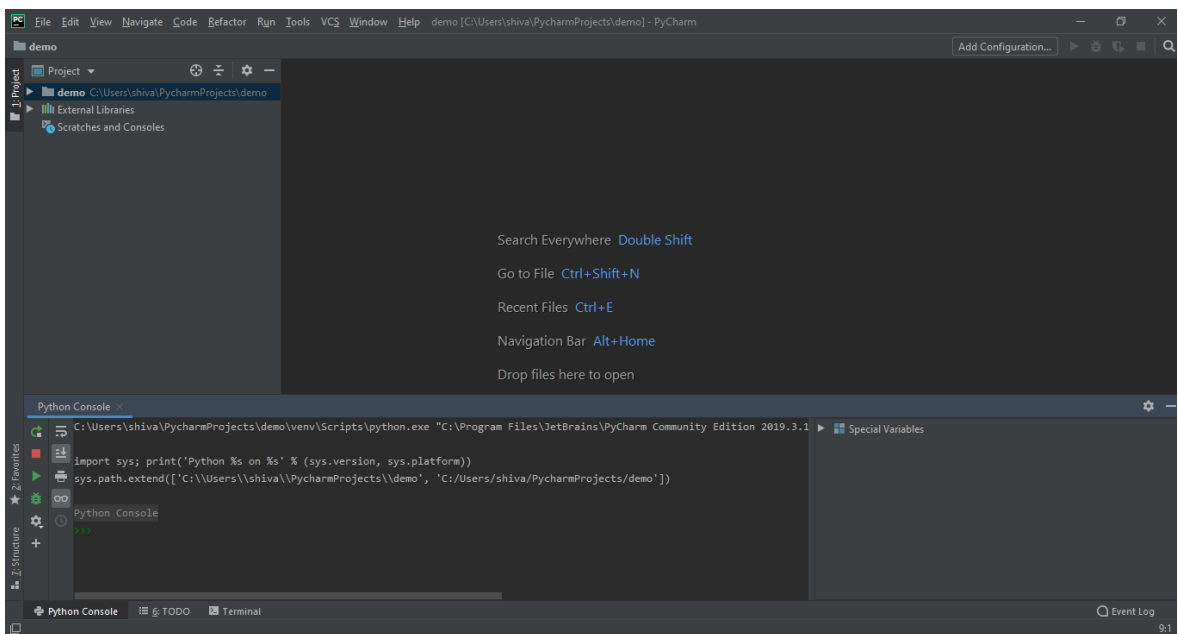


Figure.15 PyCharm Community Edition 2019.3.1

4.2.4 Installing JAVA and IntelliJ IDEA(IDE)

JAVA is available for different operating system such as Windows, Mac, Linux, Solaris. Installing JAVA8 requires administrator access to windows on host computer. And windows system requirements mentioned below.

RAM: 128 MB

Disk-Space: 124 MB for JRE; 2 MB for JAVA update

Processor: Minimum Pentium2 266mhz processor

Browsers: Internet explorer 9 and above, Firefox

Operating System: Windows 10(8u51 and above)

Download JAVA software for windows offline available on <https://java.com/en/download/manual.jsp>. The File Download dialog box appears prompting to run or save the download file Click Save to download the file to locally on host system. Close all applications including the browser. Double-click on the saved file to start the installation process. The installation process starts. Click the Install button to accept the license terms and to continue with the installation. After the installation is finished, we can verify it through Programs and Features in a control panel.

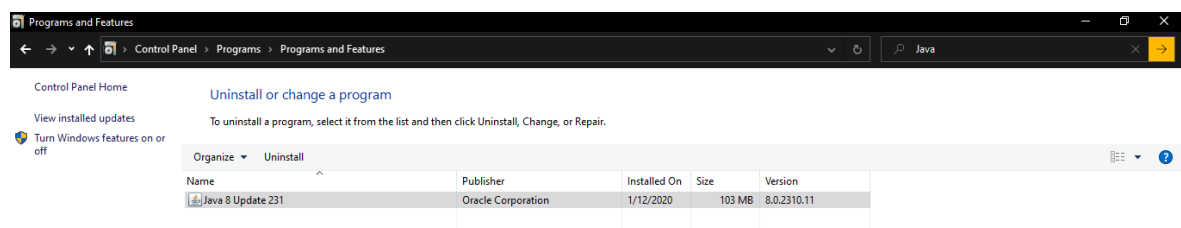


Figure.16 JAVA 8 installation

IntelliJ IDEA is a cross-platform IDE that offers consistent functionality on operating systems such as Windows, macOS and Linux. The following versions are available in IntelliJ IDEA; (i) Community Edition is open source and free, licensed under Apache 2.0. It provides all the basic features required to build JVM and Android. (ii) IntelliJ IDEA Ultimate is commercial and is sold with a trial duration of 30 days. It offers additional Web and business technology software and functionality. No need to install Java to run IntelliJ Concept, because JetBrains Runtime is bundled with the IDE (JRE 11-

based). However, a standalone JDK is required for developing Java applications. System requirements for IntelliJ IDEA are same as mentioned in Table.9 . For standalone installation of IntelliJ IDEA available on <https://www.jetbrains.com/idea/download/#section=windows>. we can save the .exe for community version which is open source below mentioned.

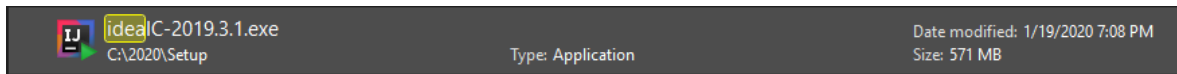


Figure.17 Executable files-IntelliJ IDEA community edition

Upon completion of the installation, the host device will launch the IntelliJ IDEA IDE and request theme preferences, packages, and set up the project SDK with an integrated terminal.

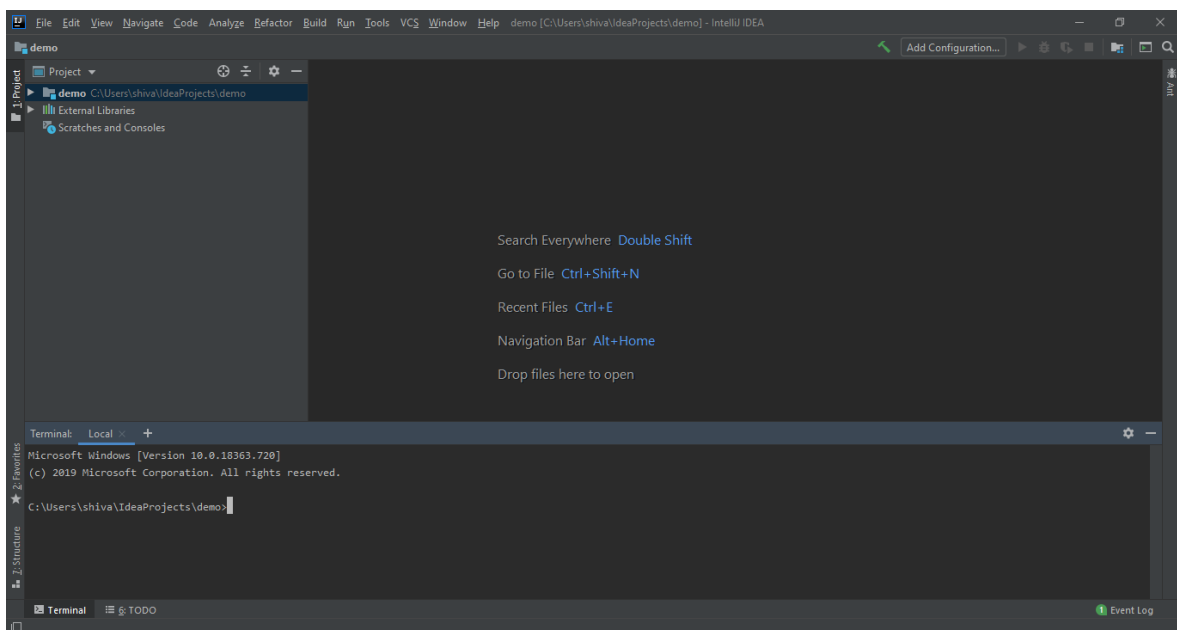


Figure.18 IntelliJ IDEA- Community Edition-2019.3.1

4.3 Code Implementation

A brief code sample in a python implemented in this section using import RE module which trying to grab a possible URL link connection with a alpha-numeric string input

provided in a python console. For comparison same logic will also deploy in a JAVA using a different packages for doing google search across the web.

4.3.1 Implementation in a Python

Browser.py has input() function which can be any alphanumeric string. And it will search through and try to find a match according to a string entered in a python console.

```
import requests
from pkg.google import search

def google():
    print("\n" "$ Please enter below First name & Last name || department, university, city,
etc.\n")

    name = input("$ Enter Here: ")
    print("\n" "$ Grab your coffee & be ready for Results..\n")

    url = "https://www.google.com/search?num=20&q=\\%s\\"

    try:
        nom2 = name.split(" ")
        name = nom2[0] + '+' + nom2[1]

    except:
        pass

    request = requests.get(url % name)
    search(r1=request)

google()
```

Figure.19 browser.py

Google.py has import the RE module and define the search function using Regular Expression which imported in browser.py to search through requested string entered in a python console.

```
import re

def search(r1=""):
    dic = {
        "%21": "!", "%23": "#", "%24": "$", "%26": "&", "%27": "'", "%28": "(", "%29": ")",
        "%2A": "*", "%2B": "+", "%2C": ",", "%2F": "/", "%3A": ":", "%3B": ";", "%3D": "=",
        "%3F": "?", "%40": "@", "%5B": "[", "%5D": "]", "%20": " ", "%22": "\"", "%25": "%",
        "%2D": "-", "%2E": ".", "%3C": "<", "%3E": ">", "%5C": "\\", "%5E": "^", "%5F": "_",
        "%60": "`", "%7B": "{", "%7C": "|", "%7D": "}", "%7E": "~", }

    info = r1.text

    urls = re.findall('url\\?q=(.*?)&', info) # Regular expression

    for url in urls:
        for char in dic:
            find = re.search(char, url)

            if find:
                decode = dic.get(char)
                url = url.replace(char, decode)

        if not "googleusercontent" in url:
            if not "/settings/ads" in url:
                if not "/policies/faq" in url:

                    print("[~] Possible Result: " + url)
```

Figure.20 Regular Expression in Google.py

RE module has search() method that can define using importing RE module in Google.py.

```
def search(pattern, string, flags=0):  
    """Scan through string looking for a match to the pattern, returning  
    a Match object, or None if no match was found."""  
    return _compile(pattern, flags).search(string)
```

Figure.21 def search() method

4.3.2 Implementation in JAVA

In 2014 Google chose to close down its search API. It's difficult to know just why. Maybe making a search API didn't make practical sense to them, or maybe it's a tool they chose not to offer anymore because it benefited SEO agencies or something in the sense that Google didn't want the data it was collected to give away.

```
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
http://maven.apache.org/xsd/maven-4.0.0.xsd">  
    <modelVersion>4.0.0</modelVersion>  
  
    <groupId>org.example</groupId>  
    <artifactId>osint</artifactId>  
    <version>1.0-SNAPSHOT</version>  
    <packaging>jar</packaging>  
  
    <properties>  
        <!-- https://maven.apache.org/general.html#encoding-warning -->  
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
        <maven.compiler.source>1.8</maven.compiler.source>  
        <maven.compiler.target>1.8</maven.compiler.target>  
    </properties>
```

```

<dependencies>
  <dependency>
    <groupId>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
    <version>1.11.3</version>
  </dependency>
</dependencies>

<build>
<plugins>
<!--
This plugin configuration will enable maven to include the project dependencies
in the produced jar file.
It also enables us to run the jar file using `java -jar command`
-->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <version>3.2.0</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        <transformers>
          <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTrans
former">
            <mainClass>pkg.Browser</mainClass>
          </transformer>
        </transformers>
      </configuration>
    </execution>
  </executions>
</plugin>
</plugins>
</build>

</project>

```

Figure.22 Maven Configuration pom.xml file

```

package pkg;

import java.util.Scanner;

public class Browser {

    private static final String HIT_URL =
        "https://www.google.com/search?num=20&q=%s";

    public static void main(String[] args) {
        google();
    }

    private static void google(){
        System.out.print("\n$ Please enter below First name & Last name" +
            " || department, university, city, etc.\n");
        System.out.print("\n$ Grab your coffee & be ready for Results...\n");
        System.out.print("\n$ Enter text: ");
        Scanner input = new Scanner(System.in);
        String receivedInput = "";
        receivedInput += input.nextLine();
        input.close();

        try {
            String[] inputArray = receivedInput.split(" ");
            if (inputArray.length > 1) {
                receivedInput = inputArray[0] + "+" + inputArray[1];
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        String queryURL = String.format(HIT_URL, receivedInput);
        new Google().search(queryURL);
    }
}

```

Figure.23 Browser.java

JSoup is a simple open-source library which provides very convenient data extraction and manipulation functionality by using DOM traversal or CSS selectors to find data. It does not support parsing based on X-Path.

```

package pkg;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Google {

    private static Map<String, String> dic = getMap();

    public void search(String queryURL) {

        try {
            Document doc = Jsoup.connect(queryURL).get();
            List<String> urls = new ArrayList<>();
            for (Element result : doc.select("div.r a")) {
                final String url = result.attr("href");

                if (isValidExpression(url)) {
                    String replaced = replace(url);
                    if (!replaced.isEmpty()) {
                        urls.add(url);
                        System.out.println(url);
                    }
                }
            }
        }
    }
}

```

Figure.24 Google.java_1

```

    if (urls.size() <= 0){
        System.out.println("No results found!");
    }
} catch (IOException e) {
    e.printStackTrace();
    System.out.println("Some Error Occurred!");
}
}

private static boolean isValidExpression(String url) {
    try {
        new URL(url).toURI();
        return true;
    } catch (URISyntaxException | MalformedURLException e) {
        return false;
    }
}

private static String replace(String url) {
    for (Map.Entry<String, String> entry : dic.entrySet()) {
        Pattern pattern = Pattern.compile(entry.getKey());
        Matcher matcher = pattern.matcher(url);
        boolean b = matcher.find(0);
        if (b) {
            url = url.replace(entry.getKey(), dic.get(entry.getKey()));
        }
    }
}

if (!url.contains("googleusercontent")) {
    if (!url.contains("/settings/ads")) {
        if (!url.contains("/policies/faq")) {
            if (!url.contains("translate")) {
                return url;
            }
        }
    }
}
}

return "";
}
}

```

Figure.25 Google.java_2

```

private static Map<String, String> getMap() {
    Map<String, String> dic = new HashMap<>();
    dic.put("%21", "!");dic.put("%23", "#");dic.put("%24", "$");
    dic.put("%26", "&");dic.put("%27", "");dic.put("%28", "(");
    dic.put("%29", ")");dic.put("%2A", "*");dic.put("%2B", "+");
    dic.put("%2C", ",");dic.put("%2F", "/");dic.put("%3A", ":");
    dic.put("%3B", ";");dic.put("%3D", "=");dic.put("%3F", "?");
    dic.put("%40", "@");dic.put("%5B", "[");dic.put("%5D", "]"");
    dic.put("%20", " ");dic.put("%22", "\"");dic.put("%25", "%");
    dic.put("%2D", "-");dic.put("%2E", ".");dic.put("%3C", "<");
    dic.put("%3E", ">");dic.put("%5C", "\\");dic.put("%5E", "^");
    dic.put("%5F", "_");dic.put("%60", "`");dic.put("%7B", "{");
    dic.put("%7C", "|");dic.put("%7D", "}");dic.put("%7E", "~");
    return dic;
}
}

```

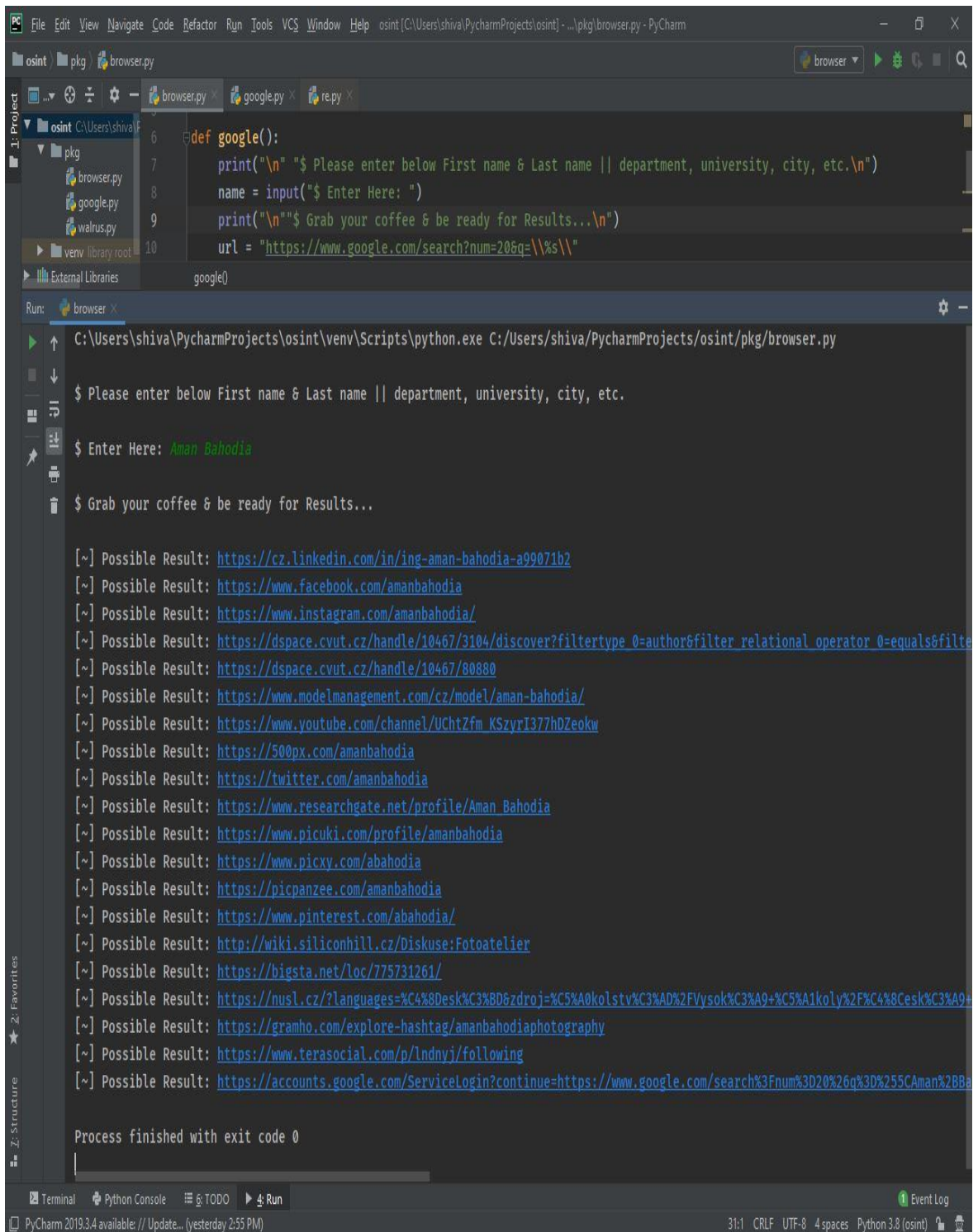
Figure.26 Google.java_3

Java.net package represents a Network Interface address. JarURLConnection a URL Connection to a Java Archive (JAR) file or an entry in a JAR file. Java.util Contains the structure of collections, classes of existing sets, pattern of events, date and time services, internationalization and various function classes (a string tokenizer, a random number generator, and a bit array). Java.io Provides input and output through data streams, serialization, and the file system.

4.4 Accomplishment

Previously, had to manually open the page on a browser to retrieve data from a website, and use copy and paste feature. This strategy works but it's one downside is that if the number of websites is large, or there is massive detail, it can get exhausting. Compared to manually copying and pasting the data into the web browser, the time taken to retrieve information from a given source is greatly decreased. The derived data is more reliable and accurately structured ensuring accuracy.

4.4.1 Result from String[a-z,A-Z]



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help osint (C:\Users\shiva\PycharmProjects\osint) - ... \pkg\browser.py - PyCharm
osint \ pkg \ browser.py
Project
osint C:\Users\shiva\F
  pkg
  browser.py
  google.py
  walrus.py
  venv library root
External Libraries
google()
Run: browser
C:\Users\shiva\PycharmProjects\osint\venv\Scripts\python.exe C:/Users/shiva/PycharmProjects/osint/pkg/browser.py
$ Please enter below First name & Last name || department, university, city, etc.
$ Enter Here: Aman Bahodia
$ Grab your coffee & be ready for Results...

[~] Possible Result: https://cz.linkedin.com/in/ing-aman-bahodia-a99071b2
[~] Possible Result: https://www.facebook.com/amanbahodia
[~] Possible Result: https://www.instagram.com/amanbahodia/
[~] Possible Result: https://dspace.cvut.cz/handle/10467/3104/discover?filtertype\_0=author&filter\_relational\_operator\_0=equals&filter
[~] Possible Result: https://dspace.cvut.cz/handle/10467/80880
[~] Possible Result: https://www.modelmanagement.com/cz/model/aman-bahodia/
[~] Possible Result: https://www.youtube.com/channel/UChTZfm\_KSzyrI377hDZeokw
[~] Possible Result: https://500px.com/amanbahodia
[~] Possible Result: https://twitter.com/amanbahodia
[~] Possible Result: https://www.researchgate.net/profile/Aman\_Bahodia
[~] Possible Result: https://www.picuki.com/profile/amanbahodia
[~] Possible Result: https://www.picxy.com/abahodia
[~] Possible Result: https://picpanzee.com/amanbahodia
[~] Possible Result: https://www.pinterest.com/abahodia/
[~] Possible Result: http://wiki.siliconhill.cz/Diskuse:Fotoatelier
[~] Possible Result: https://bigsta.net/loc/775731261/
[~] Possible Result: https://nusl.cz/?languages=%C4%8Desk%C3%BD6zdroj=%C5%A0kolstv%C3%AD%2FVysok%C3%A9+%C5%A1koly%2F%C4%8Cesk%C3%A9+
[~] Possible Result: https://gramho.com/explore-hashtag/amanbahodiaphotography
[~] Possible Result: https://www.terasocial.com/p/lndnyj/following
[~] Possible Result: https://accounts.google.com/ServiceLogin?continue=https://www.google.com/search%3Fnum%3D20%26q%3D%255CAman%2BBa

Process finished with exit code 0
Terminal Python Console TODO Run
PyCharm 2019.3.4 available // Update... (yesterday 2:55 PM) 31:1 CRLF UTF-8 4 spaces Python 3.8 (osint) Event Log
```

Figure.27 Build result of browser.py(string contain[a-z,A-Z])

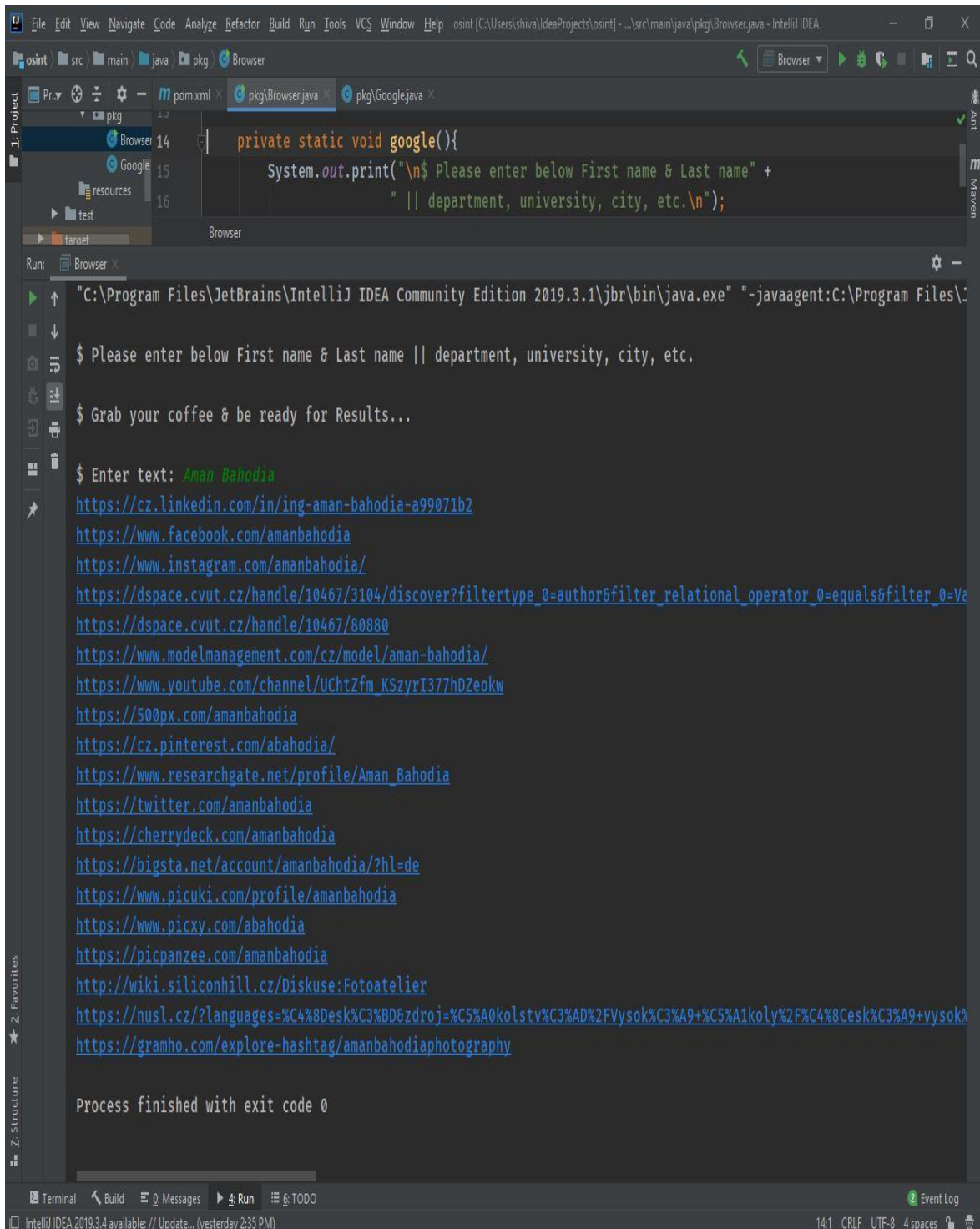
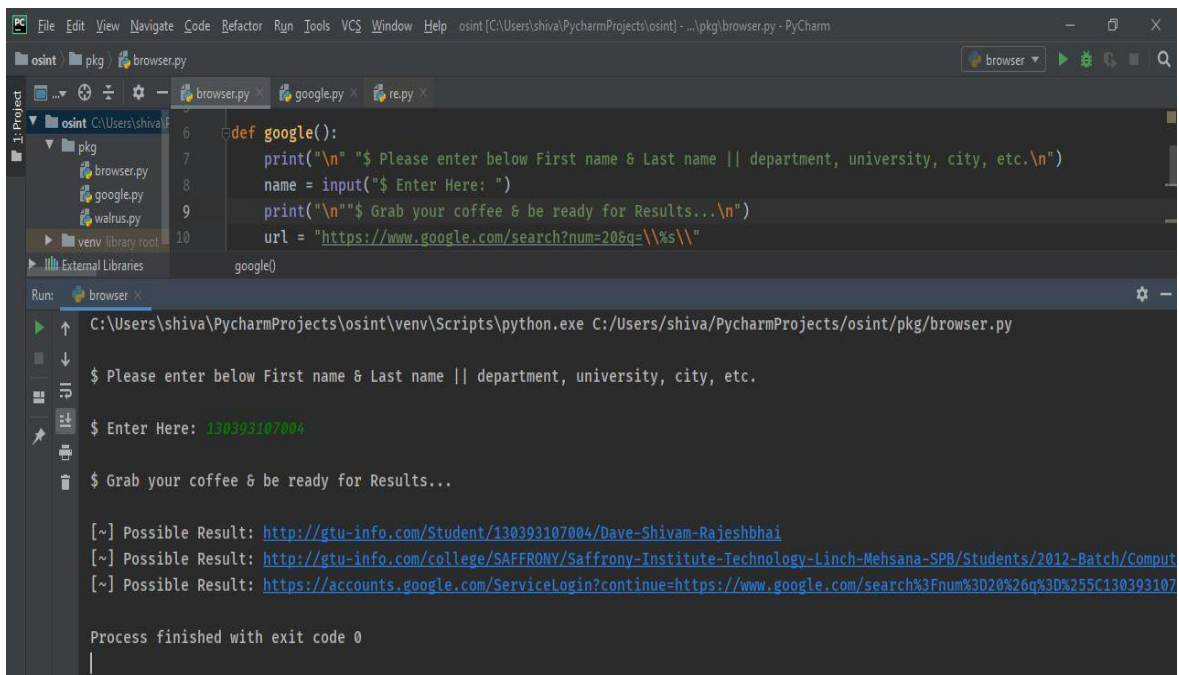


Figure.28 Build result of Browser.java(string contain[a-z,A-Z])

4.4.2 Result from Numbers[0-9]



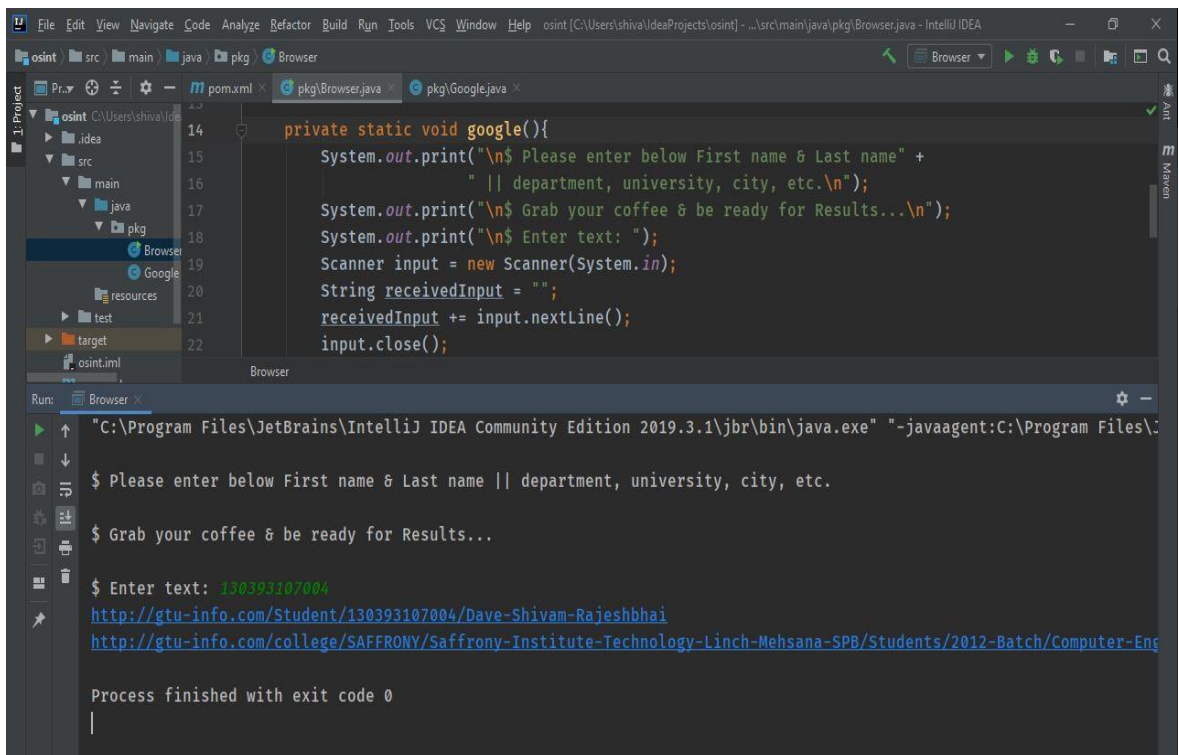
```
def google():
    print("\n" "$ Please enter below First name & Last name || department, university, city, etc.\n")
    name = input("$ Enter Here: ")
    print("\n"$ Grab your coffee & be ready for Results...\n")
    url = "https://www.google.com/search?num=20&q=\s"
```

```
Run: browser
C:\Users\shiva\PycharmProjects\osint\venv\Scripts\python.exe C:/Users/shiva/PycharmProjects/osint/pkg/browser.py
$ Please enter below First name & Last name || department, university, city, etc.
$ Enter Here: 130393107004
$ Grab your coffee & be ready for Results...

[~] Possible Result: http://gtu-info.com/Student/130393107004/Dave-Shivam-Rajeshbhai
[~] Possible Result: http://gtu-info.com/college/SAFFRONY/Saffrony-Institute-Technology-Linch-Mehsana-SPB/Students/2012-Batch/Comput
[~] Possible Result: https://accounts.google.com/ServiceLogin?continue=https://www.google.com/search%3Fnum%3D20%26q%3D%255C130393107

Process finished with exit code 0
```

Figure.29 Build result of browser.py(string contain[0-9])



```
private static void google(){
    System.out.print("\n$ Please enter below First name & Last name" +
        " || department, university, city, etc.\n");
    System.out.print("\n$ Grab your coffee & be ready for Results...\n");
    System.out.print("\n$ Enter text: ");
    Scanner input = new Scanner(System.in);
    String receivedInput = "";
    receivedInput += input.nextLine();
    input.close();
}
```

```
Run: Browser
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\jbr\bin\java.exe" "-javaagent:c:\Program Files\
$ Please enter below First name & Last name || department, university, city, etc.
$ Grab your coffee & be ready for Results...
$ Enter text: 130393107004
http://gtu-info.com/Student/130393107004/Dave-Shivam-Rajeshbhai
http://gtu-info.com/college/SAFFRONY/Saffrony-Institute-Technology-Linch-Mehsana-SPB/Students/2012-Batch/Computer-Eng

Process finished with exit code 0
```

Figure.30 Build result of Browser.java(string contain[0-9])

4.4.3 Result in Google's search engine

By clicking on each link, it can open webpage in a web browser which precisely match through Regular expression's engine.

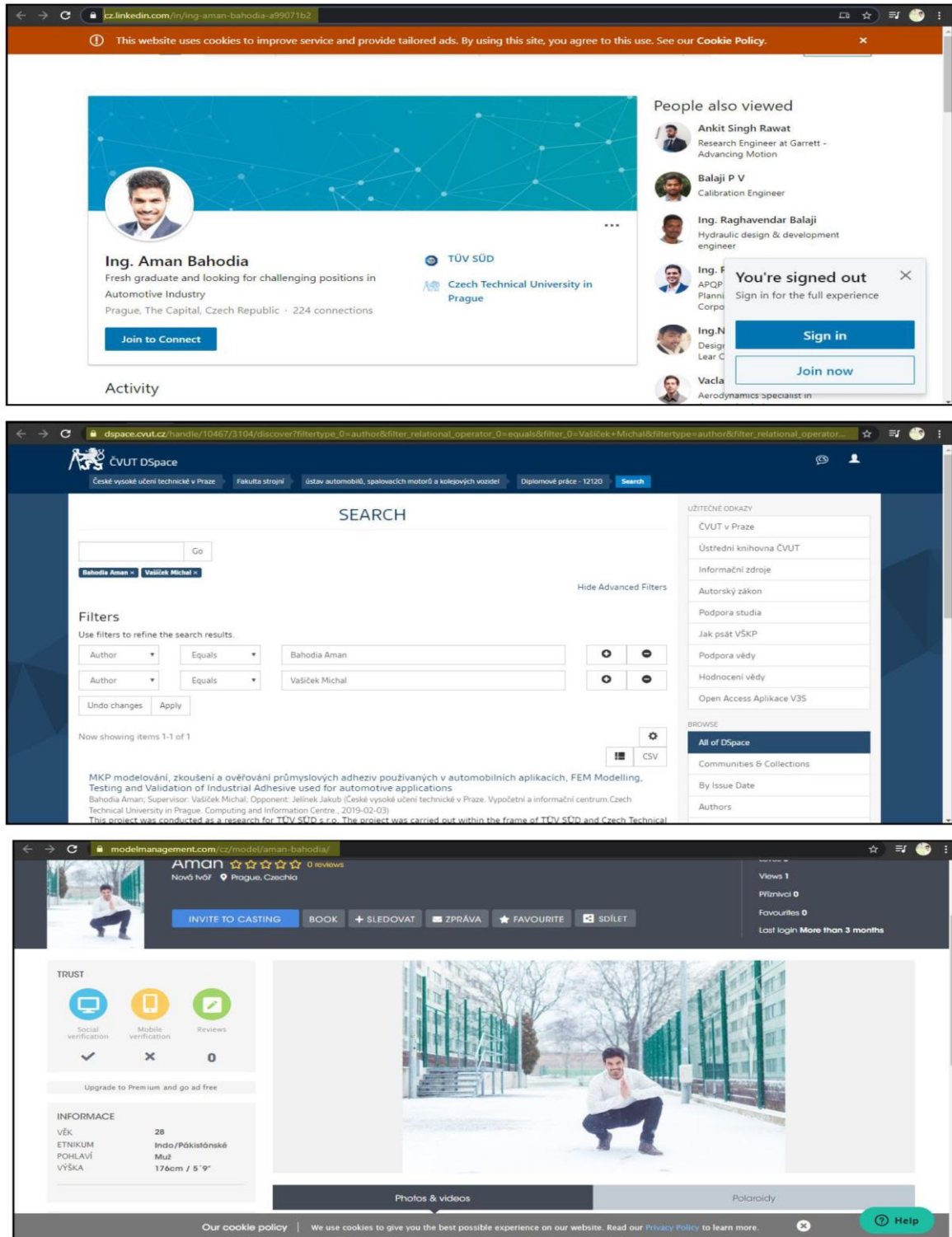


Figure. 31 Result collage_1 String[a-z,A-Z]

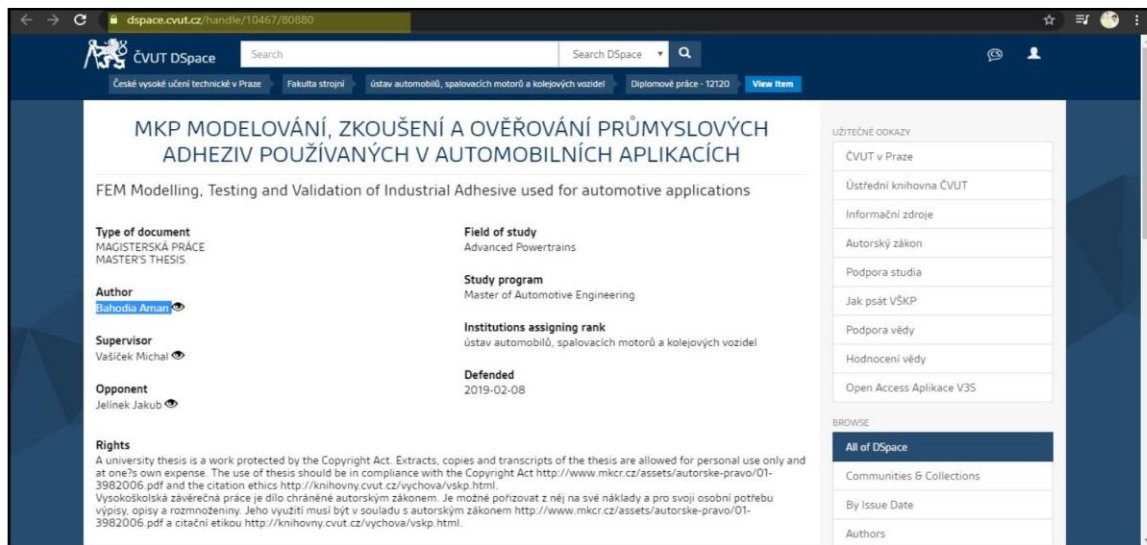
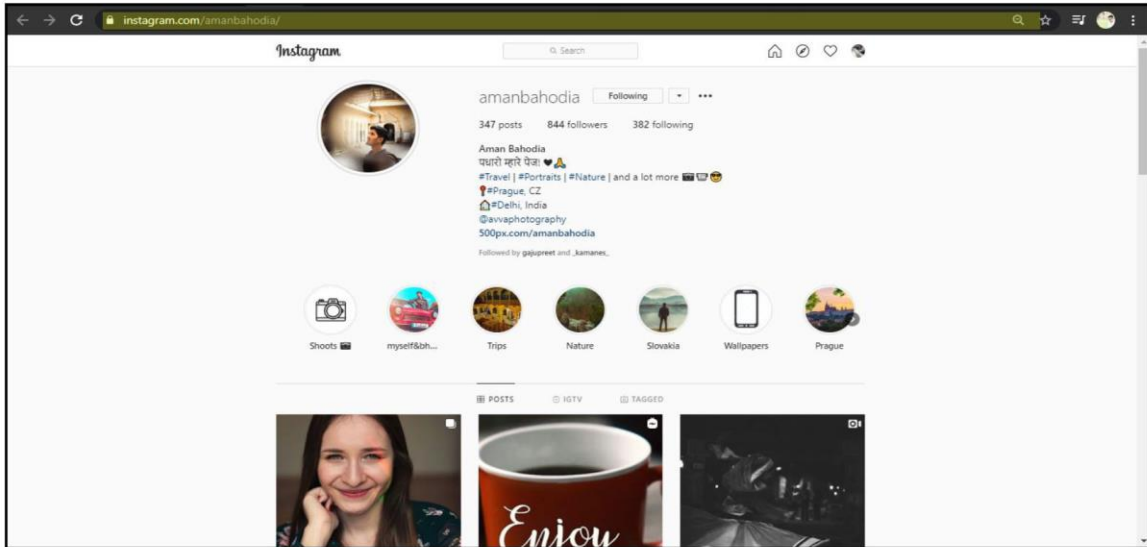


Figure.32 Result Collage_2 String[a-z,A-Z]

issuu.com/mreifaridabad/docs/degree-detail-of-fet

49	FET/AUI(F)/037	09010030357	aman bahodia			Bachelor of Technology (Automobile Engineering and Technology)
50	FET/AUI(F)/038	09010030358				Bachelor of Technology (Automobile Engineering and Technology)
51	FET/AUI(F)/039	09010030391	SUMIT	MAHINDER SINGH		Bachelor of Technology (Automobile Engineering and Technology)
52	FET/AUI(F)/040	09010030401	SUNNY VERMA	D C VERMA		Bachelor of Technology (Automobile Engineering and Technology)
53	FET/AUI(F)/041	09010030411	VARUN SHARMA	VUJAY KUMAR SHARMA		Bachelor of Technology (Automobile Engineering and Technology)
54	FET/AUI(F)/043	09010030431	VIPUL SINGLA	RAJ KUMAR		Bachelor of Technology (Automobile Engineering and Technology)
55	FET/AUI(F)/044	09010030441	AKSHAY KATHURIA	VINESH KATHURIA		Bachelor of Technology (Automobile Engineering and Technology)
56	FET/AUI(F)/045	09010030451	MILAN SHEORAN	S K SHEORAN		Bachelor of Technology (Automobile Engineering and Technology)
57	FET/AUI(F)/046	09010030461	GAURAV SINGLA	SATISH SINGLA		Bachelor of Technology (Automobile Engineering and Technology)
58	FET/AUI(F)/047	09010030471	KASHIF YOUSUF	MOHD. YOUSUF PRABHUNATH TIWARI		Bachelor of Technology (Automobile Engineering and Technology)
59	FET/AUI(F)/048	09010030481	PRATEEK TIWARI	PRABHUNATH TIWARI		Bachelor of Technology (Automobile Engineering and Technology)
60	FET/AUI(S)/001	09004100011	ABHINAV RAJPUT	AJAY KUMAR RAJPUT		Bachelor of Technology (Automobile Engineering and Technology)
61	FET/AUI(S)/002	09004100021	ABHISHEK KUMAR	JAGDISH THAKUR		Bachelor of Technology (Automobile Engineering and Technology)
62	FET/AUI(S)/003	09004100031	AFROZ AHMED	NISAR AHMED		Bachelor of Technology (Automobile Engineering and Technology)
63	FET/AUI(S)/004	09004100041	AMAN AGHA	AGHA KHAN		Bachelor of Technology (Automobile Engineering and Technology)
64	FET/AUI(S)/005	09004100051	AMAN BAHODIA	M C BAHODIA		Bachelor of Technology (Automobile Engineering and Technology)

24-25 / 84

SHARE SAVE LIKE DOWNLOAD

degree-detail-of-fet
Published on Dec 9, 2014

EXOTIC FRUITS
Asda Good Living Magazine May 2019
Exotic Fruits by Asda [Go To Magazine](#)

nust.cz/?languages=cesky&zdroj=Skolstv%2FVysoke+skoly%2FCeske+vysoke+uceni+Technicke++Praze&formatdokumentu=NE&idnuhdokumentu%5B%5D=Vysokol...

standard driving test. As different standards exist according to the country in common to test vehicle on ...

Achtenova Gabriela; Wach Maxime; Blak Martin
Ceske vysoke uceni technicke v Praze, 2019

1 Controller for Steering of Smart Dolly
Controller for Steering of Smart Dolly
MAE supervisor: Kaloše Abhishek Ravikumar; MAE opponent: Ceske vysoke uceni technicke v Praze, 2019

1 MKP modelování, zkoušení a ověřování průmyslových adheziv používaných v automobilních aplikacích
This project was conducted as a research for TÜV SÜD s.r.o. The project was carried out within the frame of TÜV SÜD and Czech Technical University in Prague. The purpose of this report is to present ...
Vašček Michal; Bahoda Aman; Jelínek Jakub
Ceske vysoke uceni technicke v Praze, 2019

1 Development of GT-Power Model of a Gasoline Engine with Low Pressure EGR Suitable for Model-Based Predictive Control of an Air Path
To develop an engine complying emission limits, reduce fuel consumption and enhance engine power output is a car manufacturer's task of the last years. New emission limits make this task even more ...
Macek Jan; Jaros Jakub; Dolecek Vit
Ceske vysoke uceni technicke v Praze, 2019

gramho.com/explore-hashtag/amanbahodiaphotography

#amanbahodiaphotography Instagram Posts
81 posts

At the highest point of Czech Republic 🇨🇪👥...
#czechrepublic #snezka #mountains #hiking #nikonphotography #amanbahodiaphotography #nature

Prejděte k T-Mobilu
T-Mobile
Poradíme Vám, jak co nejvíce ušetřit při nákupu tarifu i telefonu.
OTEVŘÍT

One winter shot by @amanbahodia ❤️
#amanbahodiaphotography #winterland #greensyes #photography #nikonphotography

Figure.33 Result Collage_3 String[a-z,A-Z]

Enrollment No.	Name	CGPA	Percentage	Backlogs
120390107049	Patel Nikita Gajendra	5.44	5.85	1
120390107055	Khanduri Dristi Harishchandra	6.12	6.41	1
130393107001	Bhavsar Kaivan Rakeshbhai	4.23	4.41	6
130393107002	Chauhan Manthansinh Jayadipsinh	4.17	4.38	7
130393107003	Dangi Vatsal Bhavinbhai	5.88	6.06	3
130393107004	Dave Shivam Rajeshbhai	6.12	6.18	0
130393107005	Gajjar Bijalben Ambalal	5.53	5.75	3
130393107006	Hathaliya Shyam Dipeshbhai	7.18	7.64	0

Figure.34 Result_4 String[0-9]

Name : DAVE SHIVAM RAJESHBAI
 Enrollment No : 130393107004
 Institute : 039--S. P. B. PATEL ENGINEERING COLLEGE, MEHSANA
 Branch : BE-07-COMPUTER ENGINEERING

ENROLLMENT NO.	CPI	CGPA	TOT BACKL	BCK1	BCK2	BCK3	BCK4	BCK5	BCK6	BCK7	BI
130393107004	6.12	6.18	0	-	-	0	0	0	0	0	

Last Exam	Sem	Declaration Date	Cur Back	SPI	Convo Year	Class	Percenta
BE SEM 8 - Regular (MAY 2016)	8	28-6-2016	0	7.40	2016	Second class	56.80

SEM 3

WINTER - 2013 (SEAT NO : E333601) CUR. BACKLO

Figure.35 Result_5 String [0-9]

4.5 Profiling Comparison

Profiling technique in python discussed in a sections 3.5 as PyCharm community edition does not integrate inbuilt version for profiling in python this feature is exclusively for commercial version, so overcome this hurdle Atom text editor and cmd is used to perform profiling.

```
import cProfile, pstats, io
import requests
import re
from pstats import SortKey

# Search() function from google.py
def search(r1=''): ...

# google() function from browser.py
def google(): ...

# cProfile module's function defined below
pr = cProfile.Profile()
pr.enable()

google()    # Function_name

pr.disable()
s = io.StringIO()
sortby = SortKey.CUMULATIVE
ps = pstats.Stats(pr, stream=s).sort_stats(sortby)
print("\n")
ps.print_stats(10) # only prints the 10 most significant lines
print(s.getvalue())
```

Figure.36 Profiling in Python


```

C:\Users\shiva\OneDrive\Desktop\py4e\kesy>profiling.py

$ Please enter below First name & Last name || department, university, city, etc.

$ Enter Here: Aman Bahodia

$ Grab your coffee & be ready for Results...

[~] Possible Result: https://www.google.com/preferences?hl=cs
[~] Possible Result: https://cz.linkedin.com/in/ing-aman-bahodia-a99071b2
[~] Possible Result: https://www.facebook.com/amanbahodia
[~] Possible Result: https://www.instagram.com/amanbahodia/
[~] Possible Result: https://dspace.cvut.cz/handle/10467/3104/discover?filtertype_0=author&filter_relational_operator_0=equals&filter_0=Va%C5%A1%C3%AD%C4%8De
kMichal&filtertype=author&filter_relational_operator=equals&filter=Bahodia+Aman
[~] Possible Result: https://dspace.cvut.cz/handle/10467/80880
[~] Possible Result: https://www.modelmanagement.com/cz/model/aman-bahodia/
[~] Possible Result: https://www.youtube.com/channel/UChtZfm_KSzyrI377hDZeokw
[~] Possible Result: https://500px.com/amanbahodia
[~] Possible Result: https://cz.pinterest.com/abahodia/
[~] Possible Result: https://www.researchgate.net/profile/Aman_Bahodia
[~] Possible Result: https://twitter.com/amanbahodia
[~] Possible Result: https://cherrydeck.com/amanbahodia
[~] Possible Result: https://bigsta.net/account/amanbahodia/?hl=de
[~] Possible Result: https://www.picuki.com/profile/amanbahodia
[~] Possible Result: https://www.picxy.com/abahodia
[~] Possible Result: https://picpanzee.com/amanbahodia
[~] Possible Result: http://wiki.siliconhill.cz/Diskuse:Fotoatelier
[~] Possible Result: https://nusl.cz/?languages=%C4%8Desk%C3%BD&zdroj=%C5%A0kolstv%C3%AD%2Fvysok%C3%A9+%C5%A1koly%2F%C4%8Cesk%C3%A9+vysok%C3%A9+u%C4%8Den%C3%
AD+technik%C3%A9+v+Praze&formatdokumentu=NE&druhdokumentu%5B%5D=vysoko%C5%A1kolsk%C3%A9+pr%C3%A1ce%2FDiplomov%C3%A9&language=cs&offset=10&fulltext=ano&jazyk
=anglick%C3%BD
[~] Possible Result: https://gramho.com/explore-hashtag/amanbahodiaphotography
[~] Possible Result: https://accounts.google.com/ServiceLogin?continue=https://www.google.com/search%3Fnum%3D20%26q%3D%255CAman%2BBahodia%255C&hl=cs

14172 function calls (14093 primitive calls) in 5.261 seconds

Ordered by: cumulative time
List reduced from 664 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall  filename:lineno(function)
   1   0.000   0.000   5.261   5.261  C:\Users\shiva\OneDrive\Desktop\py4e\kesy\profiling.py:28(goog)
   1   4.546   4.546   4.546   4.546  {built-in method builtins.input}
   1   0.000   0.000   0.706   0.706  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\api.py:63(get)
   1   0.000   0.000   0.706   0.706  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\api.py:16(request)
   1   0.000   0.000   0.703   0.703  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\sessions.py:466(request)
   1   0.000   0.000   0.698   0.698  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\sessions.py:617(send)
  35   0.000   0.000   0.629   0.018  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\socket.py:655(readinto)
  35   0.000   0.000   0.629   0.018  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\ssl.py:1230(recv_into)
  35   0.000   0.000   0.629   0.018  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\ssl.py:1090(read)
  35   0.629   0.018   0.629   0.018  {method 'read' of '_ssl._SSLSocket' objects}

```

Figure.37 Profiling Result in CMD

```

14172 function calls (14093 primitive calls) in 5.261 seconds

Ordered by: cumulative time
List reduced from 664 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall  filename:lineno(function)
   1    0.000    0.000    5.261    5.261  C:\Users\shiva\OneDrive\Desktop\py4e\kesy\profiling.py:28(get)
   1    4.546    4.546    4.546    4.546  {built-in method builtins.input}
   1    0.000    0.000    0.706    0.706  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\api.py:63(get)
   1    0.000    0.000    0.706    0.706  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\api.py:16(request)
   1    0.000    0.000    0.703    0.703  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\sessions.py:466(request)
   1    0.000    0.000    0.698    0.698  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\sessions.py:617(send)
  35    0.000    0.000    0.629    0.018  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\socket.py:655(readinto)
  35    0.000    0.000    0.629    0.018  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\ssl.py:1230(recv_into)
  35    0.000    0.000    0.629    0.018  C:\Users\shiva\AppData\Local\Programs\Python\Python38\lib\ssl.py:1090(read)
  35    0.629    0.018    0.629    0.018  {method 'read' of '_ssl.SSLSocket' objects}

```

Figure.38 pstats.Stats Module(SortKey)

Meanwhile in JAVA, IntelliJ IDEA community version also does not integrate profiler, to mitigate this IntelliJ IDEA has large number of plugins. JProfiler is used to profile the code execution in JAVA and results are mentioned below.

```

"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.1\jbr\bin\java.exe
JProfiler> Protocol version 63
JProfiler> Java 11 detected.
JProfiler> 64-bit library
JProfiler> Listening on port: 34366.
JProfiler> Enabling native methods instrumentation.
JProfiler> Can retransform classes.
JProfiler> Can retransform any class.
JProfiler> Native library initialized
JProfiler> VM initialized
JProfiler> *****
JProfiler> WARNING: class data sharing is instable.
JProfiler> Add -Xshare:off to your VM options to switch it off.
JProfiler> *****
JProfiler> Retransforming 6 base class files.
JProfiler> Base classes instrumented.
JProfiler> Waiting for a connection from the JProfiler GUI ...
JProfiler> Using instrumentation
JProfiler> Time measurement: elapsed time
JProfiler> CPU profiling enabled

$ Please enter below First name & Last name || department, university, city, etc.
$ Grab your coffee & be ready for Results...
$ Enter text: Aman Bahodia

```

Figure.39 JProfiler IDE integration

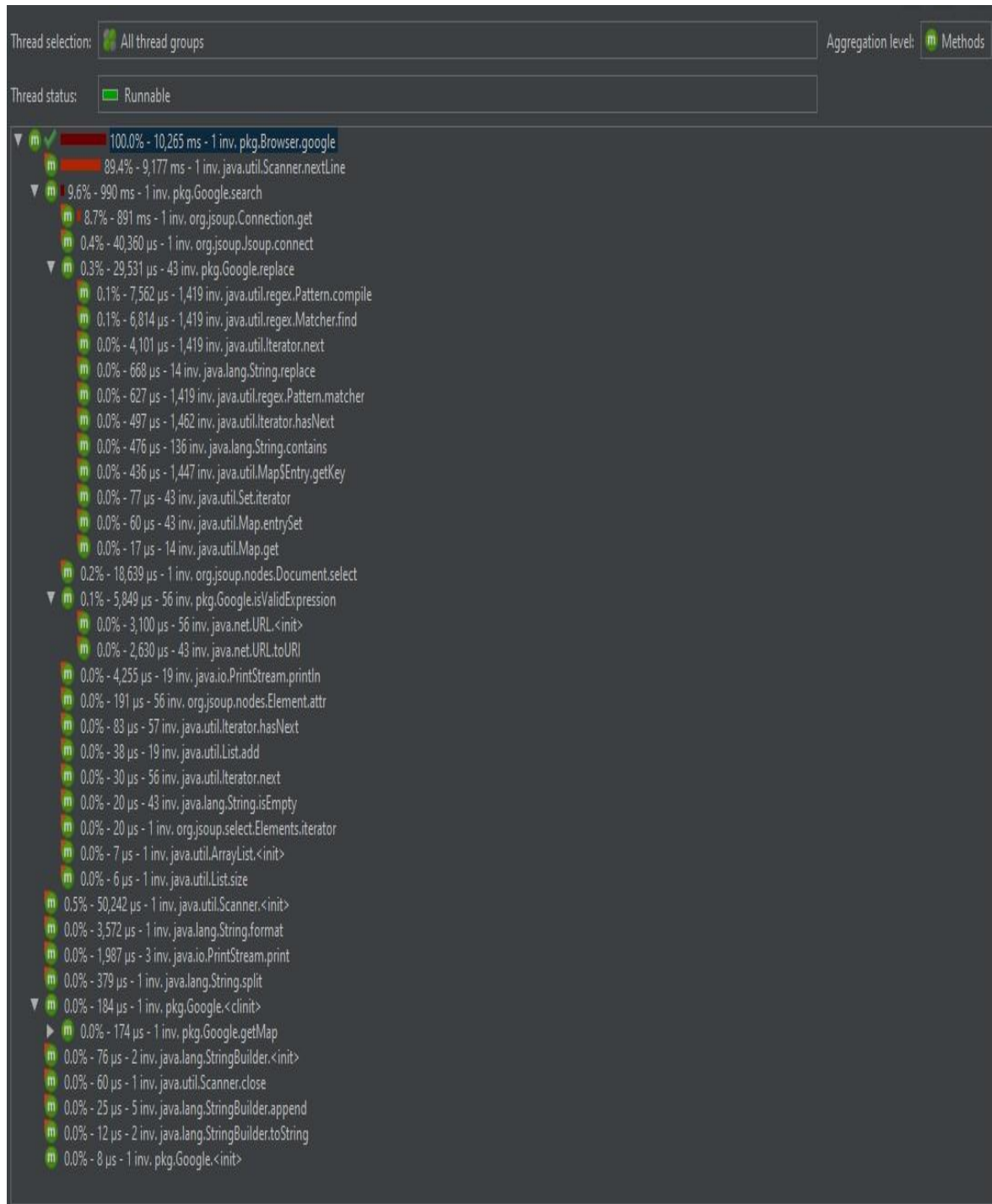


Figure.40 Jprofiler-Methods Cumulative time

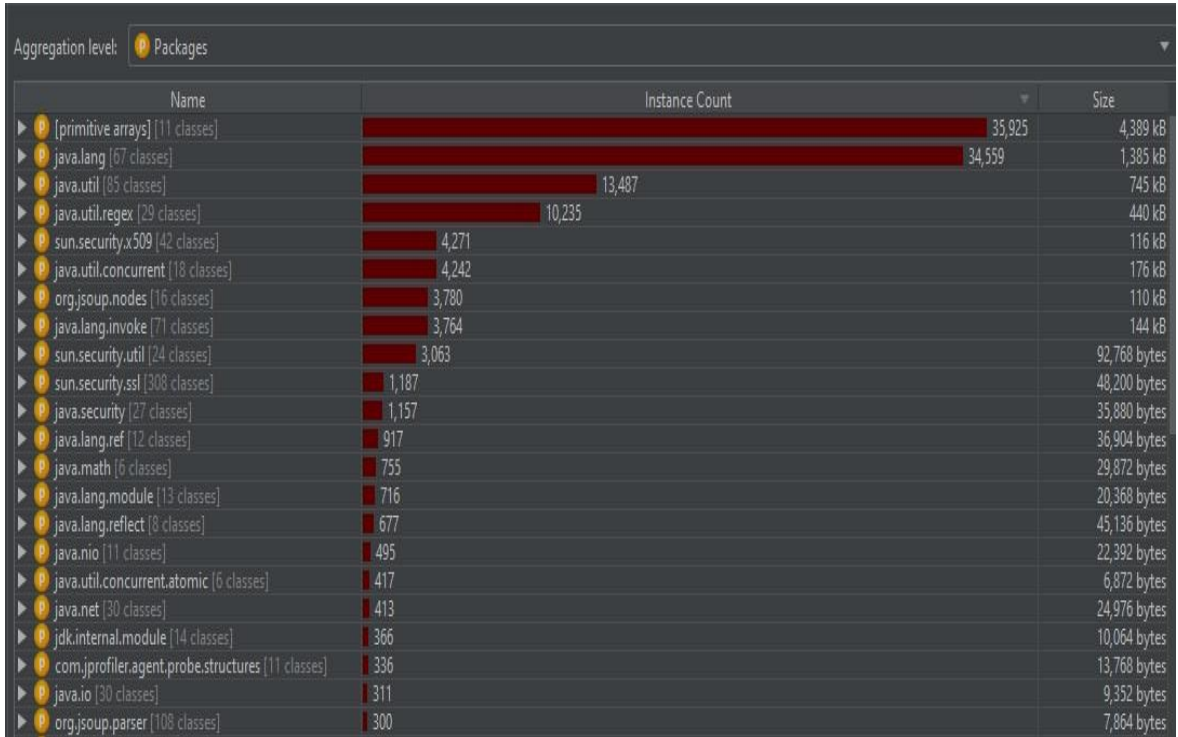


Figure.41 Jprofiler-package impact on memory

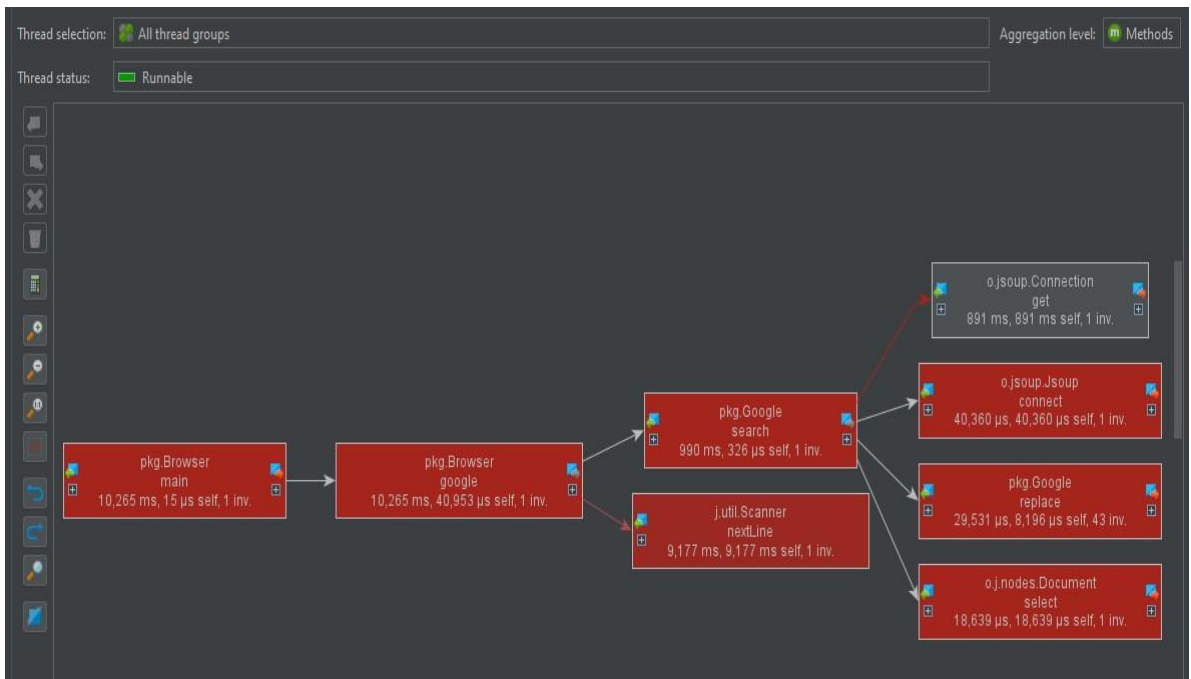


Figure.42 Jprofiler-Cumulated methods outgoing Graph

5 Results and Discussion

In the third chapter of this research document a literature review was conducted. The history of python, python data structure, operators, new features and function in python 3.8.1, python modules such as Socket with urllib package and Regular Expression, and profiling python code discussed in a depth.

In the fourth part of this research document tools and technologies were mentioned which used to achieve the objective of this study was discussed in detail. Hardware configuration of a host system and prerequisites configuration for IDEs by JetBrains were mentioned. Atom text editor was used for small implementation of new features like walrus operator, socket module and urllib package, etc.

In the practical part, deploying a logic with Regular Expression(RE) that can use google as a search engine and trying to grab a possible URL link connection with a alphanumeric input string provided in a console. Python used only two modules; requests and re to deploy a logic. While in JAVA to achieve the same result it requires JSOUP and multiple JAVA packages such as; java.io.IOException, java.net.URL, java.util.HashMap, java.util.regex.Matcher, java.util.regex.Pattern. Both code produce the same results in IDE, by clicking on a each URL it directed to browser and open a particular URL in a web browser mentioned in section 4.4.3.

Performance profiling navigate to identify bottlenecks of code, when bottlenecks identified particular portion of the code can be easily change which are creating a bottleneck. CProfile and Jprofiler were used to profile the code in Python and JAVA respectively. Terminology used in CProfile result was explained in section 3.5. Jprofiler produced graphical results as mentioned in section 4.5.

CProfile result described 14172 function calls (14093 primitive calls) in **5.261 seconds** and the table showed in cmd ordered by cumulative time. List reduced from 664 to 10 due to restriction set for ps.print_stats(10), in case to achieve full list required to eliminate 10 from the arguments can represent all 664 rows in cmd. Jprofiler produced GUI based result as mentioned in section 4.5 and it takes **10.265 seconds** to execute

pkg.Browser.google. Jprofiler produces amazing graphical statistics which can lead to investigate memory leaks and resource utilization.

5.1 Discussion

5.1.1 TIOBE Index

The TIOBE Community Software index is a measure of the programming languages popularity. The index shall be updated once a month. The scores are based on the number of professional engineers, courses, and third-party vendors worldwide. Popular search engines, including Google, Bing and Yahoo! Calculating scores, using Wikipedia, Amazon, Twitter and Baidu. It is important to remember that the TIOBE index is not about the best language for programming, or the language in which most lines of code is written.

The index can be used to test if the programming skills are still up-to-date or to make a strategic decision about what programming language should be implemented when a new software system starts to develop. The TIOBE index description can be found (**TIOBE-index**)

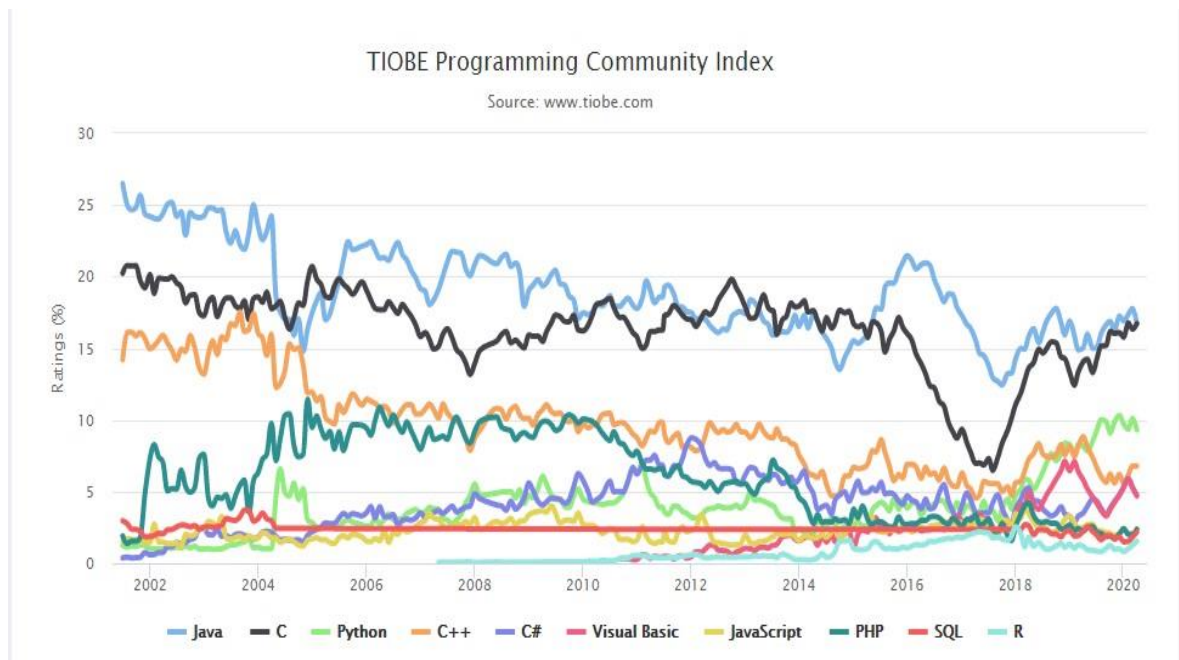


Figure.43 TIOBE Programming Community Index

6 Conclusion

The aim of this Thesis includes to understand the python as a general-purpose language which supports multiple programming paradigm such as Procedural programming, Imperative Programming, and object-oriented programming. This thesis represent the python programming language as a modern and exponentially increasing medium for application development, which in the coming future promises to be a clear rival to the Java/Javascript/PHP community. The second chapter of this study discusses in depth with the goal of this study and the methodologies used to achieve the desired results from this process.

TIOBE community announced python as the language of the year in 2007, 2010, 2018. Highest position since 2001 observed in April 2020 and ranked third as per TIOBE community. In the third chapter of this study literature review was conducted to intense analysis on Python 3.8.1. The fourth chapter of this study demonstrate the methodologies which used achieve this study's objective. Result was carried out from practical part discussed in fifth part of this study.

Python has the largest community which can contribute to the development code and it has an open source Python Package Index (PyPI) repository that provide free software developed by python community. Finally, I concluded my thesis and affirm that I have accomplished all the objectives which were assigned to me.

7 References

BEAZLEY, David M. a Brian K. JONES. 2013. *Python cookbook. 3rd.* Sebastopol: : O'Reilly, 2013. ISBN 9781449340377.

Built-in-lib. read-write-files-python. *realpython.com*. [Online]
<https://realpython.com/read-write-files-python/>.

Charles Russell Severance, Sue Blumenberg, Elliott Hauser. September 9, 2013. *Python for Everybody*. MI, USA : CreateSpace Independent Publishing Platform 7290 Investment Drive # B North Charleston SC United States, September 9, 2013. 978-1-5300-5112-0.

class-collection-deque. docs.python.org. *python.org*. [Online]
<https://docs.python.org/3/library/collections.html#deque-objects>.

DANJOU, Julien. 2019. *Serious Python: black-belt advice on deployment, scalability, testing, and more*. San Francisco : CA: No Starch Press, Inc., 2019. ISBN 1593278780;9781593278786..

David Beazley, Brian K. Jones. 2013. *Python Cookbook 3rd edition*. místo neznámé : O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2013. 978-1-449-34037-7.

dict-mapping-type. docs.python.org. *python.org*. [Online]
<https://docs.python.org/3/library/stdtypes.html#dict>.

frozenset-methods. docs.python.org. *python.org*. [Online]
<https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset>.

Indexing-lists. realpython.org/python-lists-tuples/. *RealPython*. [Online]
<https://files.realpython.com/media/t.c11ea56e8ca2.png>.

Keywords. lexical_analysis. *docs.python.org*. [Online]
https://docs.python.org/3.8/reference/lexical_analysis.html?highlight=keyword.

list-methods. docs.python.org/3/tutorial/datastructures. *python.org*. [Online]
<https://docs.python.org/3/tutorial/datastructures.html#more-on-lists>.

nestedlist-comp. docs.python.org. *python.org*. [Online]
<https://docs.python.org/3/tutorial/datastructures.html#nested-list-comprehensions>.

OSINT. Open-source intelligence. <https://en.wikipedia.org/>. [Online]

profiling. library/profile.html. <https://docs.python.org/>. [Online]
<https://docs.python.org/3/library/profile.html#instant-user-s-manual>.

PythonDOC. operator. *Python official*. [Online]
<https://docs.python.org/3/library/operator.html>.

Read&write-I/O. docs.python.org. *www.python.org*. [Online]
<https://docs.python.org/3.8/tutorial/inputoutput.html#reading-and-writing-files>.

Reconnaissance. Open-Source Intelligence. <https://medium.com/>. [Online]
<https://medium.com/@z3roTrust/open-source-intelligence-osint-reconnaissance-75edd7f7dada>.

re-module-content. docs.python.org. *www.python.org*. [Online]
<https://docs.python.org/3/library/re.html#module-contents>.

sequence-types. docs.python.org. *python.org*. [Online]
<https://docs.python.org/3/library/stdtypes.html#immutable-sequence-types>.

set-methods. docs.python.org. *python.org*. [Online]
<https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset>.

SLATKIN, Brett. 2015. *Effective Python: 59 specific ways to write better Python*.
Beaverton : Ringgold Inc., 2015. ISBN 2372-3424..

Socket-methods. socket.html. *docs.python.org*. [Online]
<https://docs.python.org/3/library/socket.html#module-contents>.

standard-type. en.wikipedia.org/Python_(programming_language). *en.wikipedia.org*.
[Online]
[https://en.wikipedia.org/wiki/Python_\(programming_language\)#/media/File:Python_3_The_standard_type_hierarchy.png](https://en.wikipedia.org/wiki/Python_(programming_language)#/media/File:Python_3_The_standard_type_hierarchy.png).

String-Methods. Python docs. *Python.org*. [Online]
<https://docs.python.org/3/library/stdtypes.html#string-methods>.

TCP-portNum. List of TCP and UDP port numbers. *Wikipedia*. [Online]
https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

TIOBE-index. tiobe-index. *www.tiobe.com*. [Online] <https://www.tiobe.com/tiobe-index/programming-languages-definition/>.