# Výuka programování v jazyce Python

**Bakalářská práce**

**Tomáš Fortelka**

**Vedoucí bakalářské práce: RNDr. Jaroslav Icha**

**Jihočeská univerzita v Českých Budějovicích**

**Pedagogická fakulta**

**Katedra Informatiky**

**2010**

# Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 10.12. 2010

Tomáš Fortelka

## Anotace

Tato bakalářská práce má za cíl vytvořit kolekci výukových lekcí, které budou využitelné pro výuku objektově orientovaného programování na střední i vysoké škole. Python je dynamický objektově orientovaný jazyk, který se kromě využití v praxi, setkává s příznivou odezvou i jako jazyk používaný v úvodních kurzech objektově orientovaného programování.



## Abstract

The target of this Bachelor thesis is to create a training-lesson's collection, which will be usable for teaching object oriented programing in high schools and also in universities. The Python language is dynamic object oriented language, which has, beside using in practice, very good response also as the language used in the preliminary courses object oriented programing.

# Poděkování

Rád bych poděkoval RNDr. Jaroslavu Ichovi za ochotu, čas, organizační a odborné vedení při zpracování této práce.

# Obsah

# 1 Úvod

Prostřednictvím této bakalářské práce bych chtěl přispět k výuce programování v jazyce Python. Po prostudování knih, zabývajících se programovacím jazykem Python, jsem se rozhodl vytvořit kolekci výukových lekcí, které by měly sloužit pro snazší pochopení jazyka Python. Pro koho je práce určena a jakým směrem je práce vypracována, vyplývá z úvodního dotazníku, který byl podán studentům JU. Po zhodnocení úvodního dotazníku vyplynulo, že studenti jsou začátečníci, zájem mají o zaměření na multimedia a elektronickou podporu práce. Vyhodnocený úvodní dotazník je v příloze bakalářské práce. Po vypracování výukových lekcí byl podán studentům JU dotazník, ve kterém vyjádřili svůj názor na výukové lekce. Vyhodnocení tohoto dotazníku je popsáno v závěru bakalářské práce.

## 1.1 Komu je práce určena

Tato bakalářská práce je určena studentům středních škol a studentům vysokých škol. Práce slouží jako doplňkový materiál při studiu.

## 1.2 Elektronická podpora práce

V systému eAMOS jsou vytvořeny výukové lekce. Každá lekce obsahuje výkladovou část a otázky vztahující se k výkladové části. V systému eAMOS jsou uloženy flashové animace, které slouží pro vizualizaci příkazu for cyklu a větvení programu.

## 1.3 Používaný software v bakalářské práci

Pro potřeby výuky jsem zvolil Netbeans. Toto vývojové prostředí jsem zvolil z důvodu větší přehlednosti kódu a dalšího možného použití vývojového

prostředí při programování v Javě, PHP. K tomuto vývojovému prostředí je k dispozici i velice dobrá dokumentace na webu.

## 1.4 Stručný popis práce

Zájemce o programovací jazyk Python může začít pracovat s touto prací bez předešlých zkušeností a vědomostí.

V práci je popsáno vytvoření nového projektu, editace projektu, spouštění projektu

V dalších kapitolách práce nalezneme datové typy, cykly, větvení programu a další základní rysy Pythonu.

V závěrečných kapitolách práce je znázorněna ukázka využití speciálních modulů pro práci s multimedii. Ukázka spočívá ve vytvořeném přehrávači videa a zvuku.

## 1.5 Typografická konvence

Z důvodu větší přehlednosti a srozumitelnosti práce jsem zvolil následující typografickou úpravu klíčových slov jazyka Python, názvy proměnných a zdrojových kódů. V Pythonu špatně funguje český jazyk. Z tohoto důvodu jsou zdrojové kódy vypsány bez diakritiky.

| | |
|---|---|
| `Zdrojovy kod` | Výpis zdrojového kódu |
| **Type, continue** | Klíčová slova jazyka Python |
| `>>> 'Bydlím ve městě'` | Vstup do interpreta |
| ***'Bydlím ve městě'*** | Výstup interpreta |

**Tabulka 1: Typografická konvence**

# 2 Analýza zdrojů

Analýza zdrojů se zabývá popisem internetových stránek a knih, které se používají při studiu programování v Pythonu. Cílem této kapitoly je získání přehledu dostupných kurzů a jejich chyb. Dalším cílem je popis nejpoužívanějších internetových stránek a knih slouží k výuce Pythonu. Z tohoto popisu by mělo být patrné, co daná internetová stránka nebo kniha obsahuje za informace. Tento popis usnadní zájemci o programování v jazyce Python rozhodování ve výběru vhodné knihy, případně internetové stránky ke studiu.

## 2.1 Analýza kurzů na internetu

Analýza kurzů na internetu byla zaměřena na kurzy, které jsou určeny jako úvodní kurzy pro studenty vysokých škol. Při vyhledávání kurzů na internetu bylo zohledněno pořadí kurzů, které byly nalezeny vyhledávači. Použil jsem vyhledávače Google, Seznam, Yahoo. Do vyhledávačů bylo zadáno „Python course“. Pro nalezení českých kurzů bylo zadáno do vyhledávače „Kurz Pythonu“. Zobrazené pořadí v tabulce 2 je vytvořeno podle pořadí vyhledávání vyhledávačů. Kurzů jsem vybral sedm, z toho pět anglicky psaných kurzů a dva české kurzy.

| Název kurzu | Verze Pythonu | Pro koho je kurz určen | Předchozí znalosti | Jazyk | Zaměření kurzu | Splnění cíle kurzu |
|---|---|---|---|---|---|---|
| A Python Course | Uvedena | Studenty vysokých škol | Neuvedeno | Angličtina | Všeobecné | Ano |
| Python short course | Neuvedena | Není specifikováno | Neuvedeno | Angličtina | Všeobecné | Ne |
| Python course in Bioinformatics | Neuvedena | Studenty bioinformatiky | Neuvedeno | Angličtina | Bioinformatika | Ne |
| Python Tutorial Zone | Neuvedena | Není specifikováno | Neuvedeno | Angličtina | Všeobecné | Ne |
| Instant Python | Neuvedena | Není specifikováno | Neuvedeno | Angličtina | Všeobecné | Ne |
| Python | Neuvedena | Není specifikováno | Neuvedeno | Čeština | Všeobecné | Ne |
| Jak se naučit programovat | Uvedena | Není specifikováno | Neuvedeno | Čeština | Všeobecné | Ne |

**Tabulka 2: Přehled kurzů a jejich zhodnocení**

Vybral jsem si několik požadavků, které by měly kurzy mít. Tyto požadavky jsou pro zájemce kurzu nutné, aby podle nich vyhodnotili, jestli jim kurz bude vyhotovovat, či nikoliv. Mezi tyto požadavky na kurz patří například verze Pythonu.

Verze Pythonu je důležitá z hlediska rozdílné syntaxe mezi verzí 2.x a verzí 3.x. Zájemce o kurz by měl vědět, v jaké verzi Pythonu je kurz prezentován. Díky této informaci lze předejít komplikacím, která může způsobit rozdílná syntaxe. Po neúspěšném zhotovení programu, jehož nefunkčnost je způsobena rozdílnou syntaxí verzí Pythonu, může vést k odrazení zájemce od kurzu a

celkově i od programování v jazyce Python. Proto je tento požadavek na kurz nutný.

Dalším velice důležitým požadavkem na kurz je kategorie, pro kterou je určen. Znalosti zájemců o kurz jsou rozmanité. Zájemce by měl vědět, zda je tento kurz určen právě pro něho. Pokud je například určen pro vysoké školy, je jasné, že se podle něho nebude učit žák základní školy.

Předchozími znalostmi se rozumí znalosti předchozích programovacích jazyků. V kurzu je nutná znalost i jiného programovacího jazyka než Pythonu. Pokud by toto hledisko nebylo uvedeno a zájemce kurzu by o tom nevěděl, mohlo by se stát, že zájemce by si nedokázal vytvořit přehled určité problematiky v programování.

Jazyk je rozhodující pro zájemce o kurz. Zájemci, kteří mají jazykovou bariéru, uvítají raději kurz ve svém mateřském jazyce.

Zaměření kurzu znamená, jaký je směr výuky programování v jazyce Python. Pro zájemce o kurz je tedy velice podstatné zaměření kurzu. Pokud tato informace chybí, nebo nevyplývá z kurzu ihned, může se zájemce část kurzu už naučit a po chvíli zjistí, že kurz má jiné zaměření než očekával.

Splnění cílů kurzu je všeobecné tvrzení. Snažil bych se toto tvrzení specifikovat blíže. Zjistil jsem, že nejčastější nesplnění cíle kurzu je způsobeno autorem kurzu. Ten si dá za cíl kurzu přehnané množství probírané látky a ve výsledném kurzu je zhruba polovina toho, co si autor sliboval. Další častou chybou je nedodržení plánované délky kurzu. Jako další příklad bych uvedl nesprávný směr kurzu. Autor kurzu si určí směr, například na grafické prostředí, a v kurzu se o něm zmiňuje pouze okrajově.

## 2.2 Popis oficiálních webů Pythonu na internetu

První stránkou, která bude asi nejčastěji navštěvovaná je www.py.cz, ta je v češtině. Na této stránce najdeme vše potřebné pro začátky programování.

Na úvodní stránce máme na výběr celý obsah tohoto webu. Na stránce se nabízí tutoriály. Tutoriály jsou velmi pěkně udělané a dobře pochopitelné.

Dále je možno vidět, jak nainstalovat Python na různé platformy. Při jakémkoliv problému s instalací doporučuji prostudovat tento web. V menu se nachází odkaz, který se zabývá problematikou češtiny pro různé verze, překlad projektu a k dispozici je i několik hotových projektů. Na stránce máme přehled editorů, které se dají použít pro programování v jazyce Python. Je to například IDLE, PSpad, VIM. Dále se tento web zabývá grafickým prostředím (GUI). Na této části webu přehledně popisují, jak a proč vybrat určitý modul pro GUI. Do podrobností nejvíce popisují modul Tkinter. Obsahem popisu modulu Tkinter je několik kapitol, jak začít s tímto modulem. Na této části webu jsou názorně zpracovány úvodní lekce v modulu Tkinter. Další část webu se zabývá modulem WxPython. Bohužel na tomto webu se o modulu moc nezmiňují, což je podle mne škoda, protože s ním mám osobně mnohem lepší zkušenosti než s Tkinterem. V následujících výukových lekcích se budeme zabývat jak Tkinterem, tak i WxPython. Na webu v záložce GUI jsou zastoupeny moduly, které se zabývají především vytvořením a následným vývojem her.

Následující část webu se zabývá databázemi a to znamená spolupráci Pythonu s databázovým systémem typu Oracle, MySQL. Tímto tématem se v našem kurzu nebudeme příliš zabývat, protože si myslím, že to patří do jiných kurzů. Následně se na tomto webu dozvíme, že se Python používá především pro práci se soubory a to jak textovými, tak i binárními. Na webu v sekci, jež je vyhraněná na práci se soubory, můžeme najít jednoduše

napsané články, které nám pomůžou pochopit problematiku. Články obsahují například praktické ukázky programu, části kódu a popis kódu.

Poté následuje sekce tipy a triky, zajímavé téma, důležité pro toho, kdo by chtěl něco vylepšit, nebo pozměnit. Následuje sekce historie Pythonu. Na konci je Pythonway, kde je napsáno, jaká mohou být jména proměnných, pasti jazyka Python. Vřele doporučuji pročíst.

Vrátíme se na hlavní stránku www.py.cz a v menu vidíme Kde začít -> Python začínáme – tady jsou popsány tutoriály, knihy, jejich ukázka a popis kódu.

## 2.3 Analýza knižních zdrojů

Na trhu je mnoho knih, které se dají použít jako příslušný zdroj informací při studiu programování. Proto jsem vybral několik knih. První kniha se jmenuje Object-Oriented Programing in Python a další velmi dobře propracovaná kniha je Introduction to Computning and Programming in Python A Multimedia Approach.

První zmíněná kniha se hodí pro začátečníka v programování v jazyce Python. Ovšem u této knihy je důležitá předchozí znalost nějakého programovacího jazyka. Tato znalost nějakého předchozího programovacího jazyka usnadní pochopení problematiky objektově orientovaného programování, kterému se kniha věnuje. Kniha je dobře čitelná a najdete v ní hodně dobrých triků. Má však jednu nevýhodu, nevěnuje se grafickému prostředí. Pro tvoření programů s grafickým prostředím je tedy nutná ještě jedna kniha, popřípadě vyhledávání kurzu na internetu.

Kniha Introduction to Computning and Programming in Python A Multimedia Approach popisuje Python pro práci s multimedii. Kniha pracuje s aplikací JES (Jython Environment for Students), která umožňuje jednodušší práci s vytvářením programu. Dále rozepisuje princip práce s multimedii na

počítačích. Kniha je rozdělena na jednotlivé sekce, které se věnují problematice obrázků, zvuku a videa. V každé takovéto sekci je mnoho ukázek kódu pro práci s danou problematikou. Na přiloženém CD jsou uloženy všechny ukázky a programy včetně instalace JES.

# 3 Seznámení s Pythonem

Python je objektově orientovaný programovací jazyk, který se může využít v mnoha oblastech vývoje softwaru. Nabízí významnou podporu k integraci s ostatními jazyky, nástroji a přichází s mnoha standardními knihovnami. Jeho použití je velice široké od programů na zpracování multimedií až po zpracování textů. Python není závislý na platformě, na které běží. Aktuální verze Pythonu je 2.6 a 3.1.

## 3.1 Historie Pythonu

Počátek Pythonu je rok 1990, kdy ho vymyslel Guido van Rossume. V roce 2001 byla založena nezisková organizace Python Software Foundation, která se zabývá vývojem tohoto jazyka.

## 3.2 Programové vybavení Pythonu

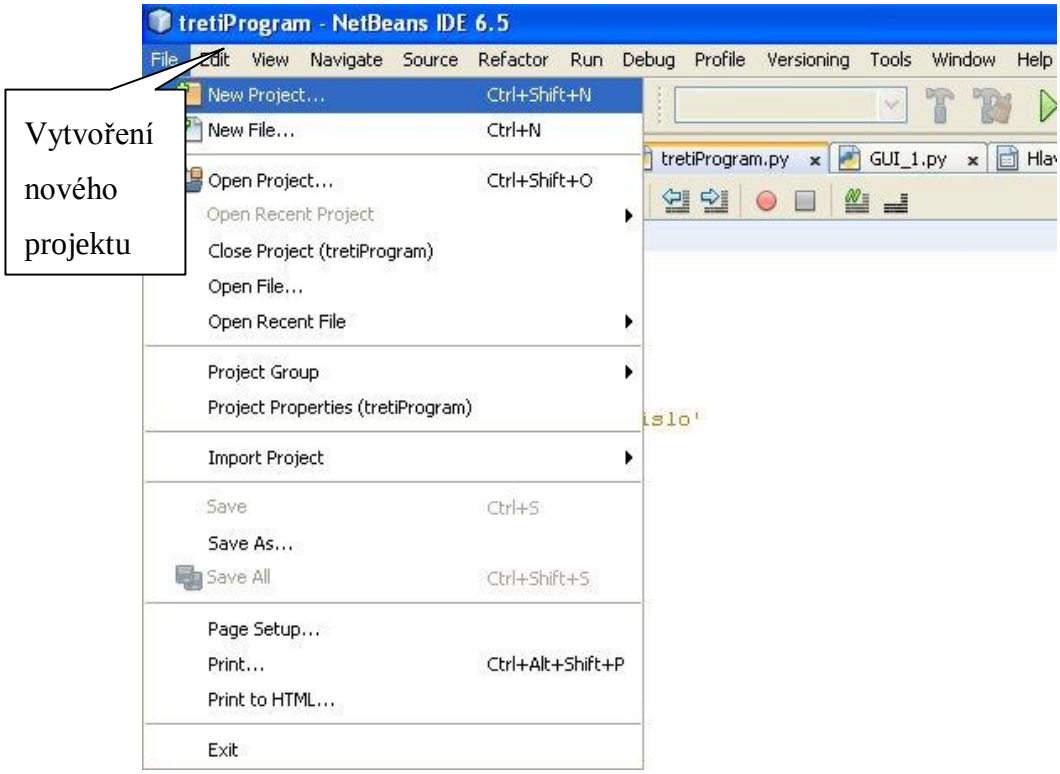
Pro studium této bakalářské práce je nutné nainstalovat příslušné softwarové vybavení.

1. **Instalace Pythonu verze 2.6**
2. **Instalace vývojového prostředí Netbeans**
3. **Konfigurace Netbeans pro práci s Pythonem**
4. **Instalace WxPython**

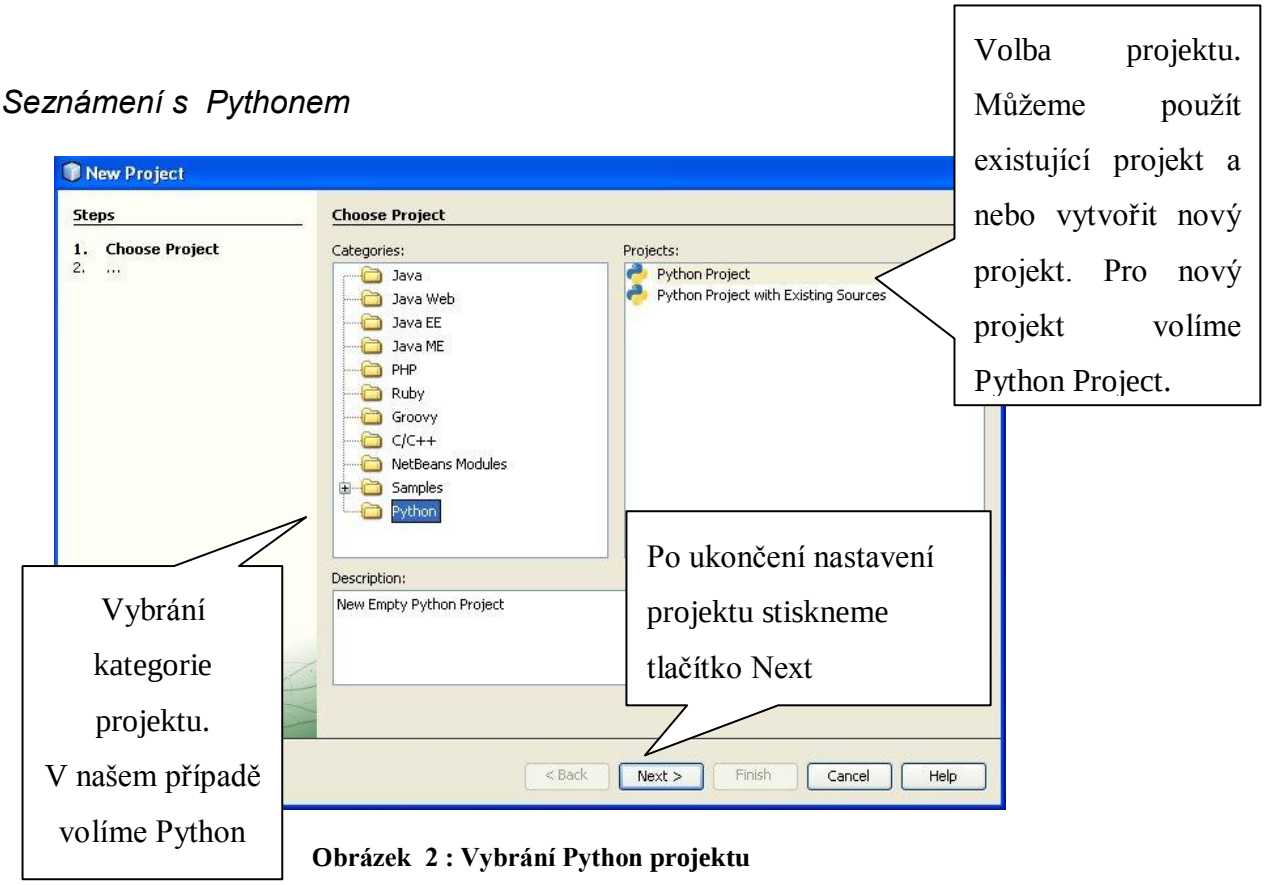
Python se dá bezplatně stáhnout na http://www.python.org/download/
Vývojové prostředí Netbeans si můžete bezplatně stáhnout na
http://netbeans.org/downloads/index.html
Modul WxPython si můžete stáhnout na
http://www.wxpython.org/download.php. Modul WxPython je nutné vybrat podle nainstalované verze Pythonu.

## 3.3 Vytvoření nového projektu

V titulkovém menu vybereme možnost File (Obrázek 1). Jak je již na obrázku patrné, zvolíme New Project, následně se nám objeví dialogové okno (Obrázek 2).

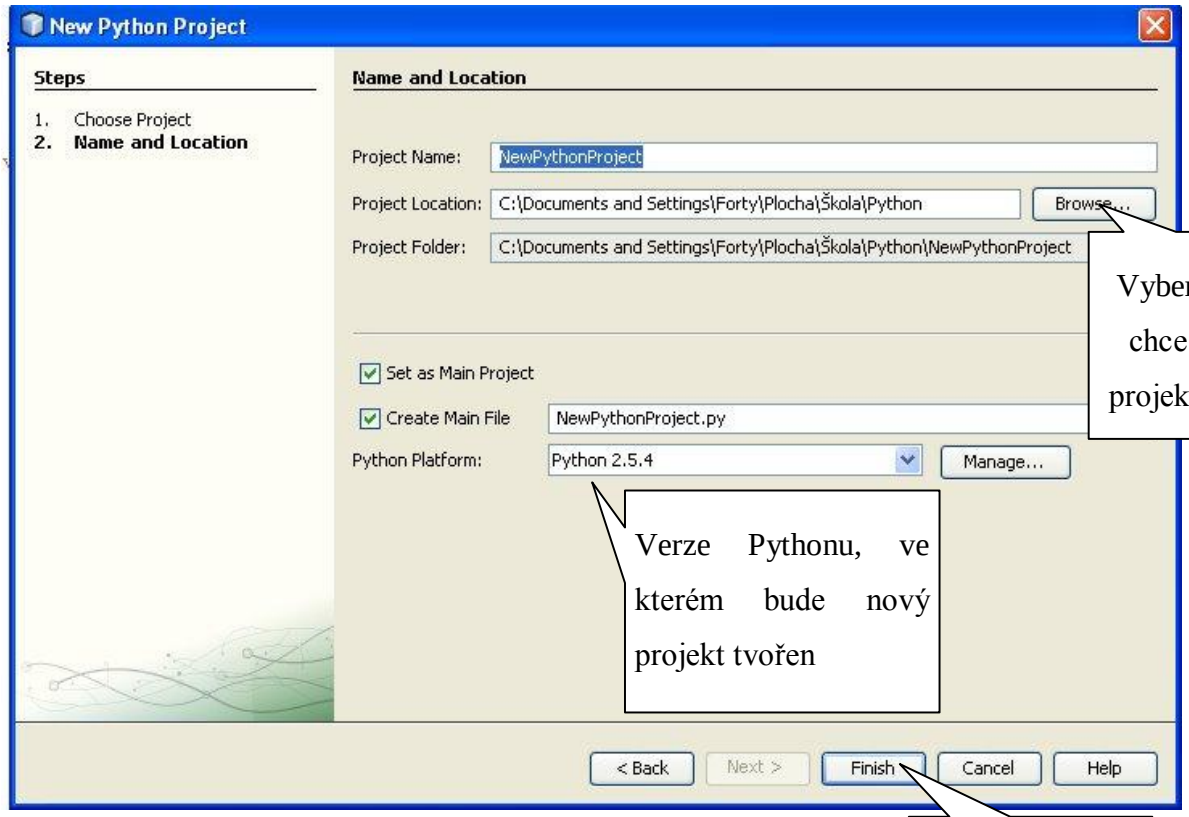



**Obrázek 1: Otevření Python projektu**




**Obrázek 2 : Vybrání Python projektu**

Po vybrání projektu podle obrázku 2 se dostáváme k dalšímu oknu průvodce vytváření projektu a to znázorňuje obrázek 3. Toto okno se jmenuje nový Python projekt a slouží pro zadání názvu projektu, kam projekt uložit a hlavně verzi Pythonu, ve které bude projekt tvořen. Více o Netbeans na stránkách: http://WIKI.NETBEANS.ORG/PYTHON.




**Obrázek 3: Zvolení a uložení Python Projektu**

Po stisknutí tlačítka Finish se nám vytvoří stránka pro psaní kódu. Netbeans vygeneruje jméno autora, datum vzniku a kód, který po spuštění vypíše „Hello World“. Vygenerovaná stránka nového projektu je zobrazena na obrázku 4.

```
1  # To change this template, choose Tools | Templates
2  # and open the template in the editor.
3
4  __author__="Forty"
5  __date__ ="$12.6.2010 11:11:18$"
6
7  if __name__ == "__main__":
8      print "Hello World"
9
```

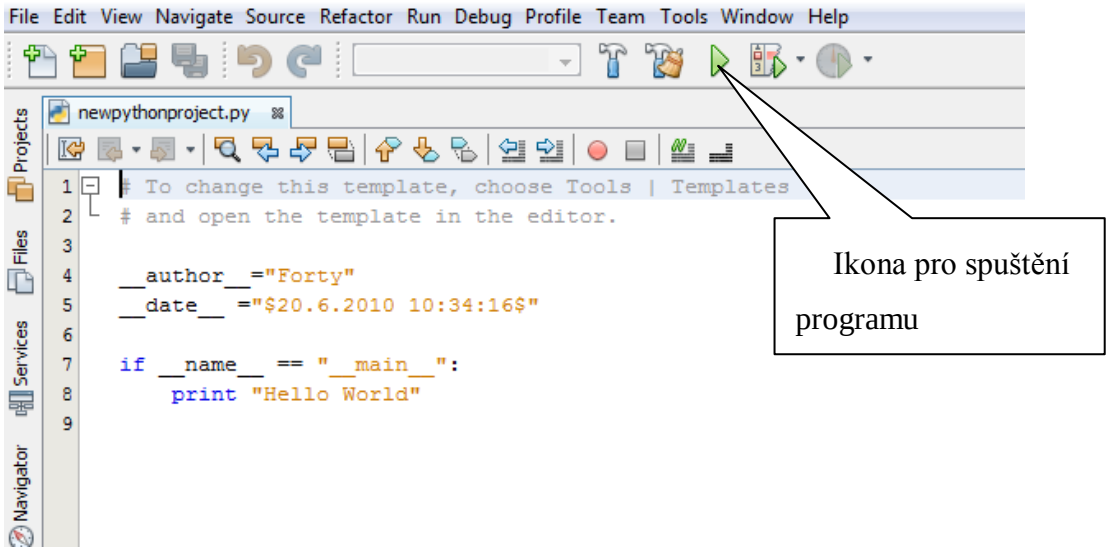
**Obrázek 4: Nový soubor se základními informacemi**

## 3.4 Editace programu

Editaci programu si vyzkoušíme na jednoduchém programu, který vypíše „Hello World“. Program je zobrazen na obrázku 4.

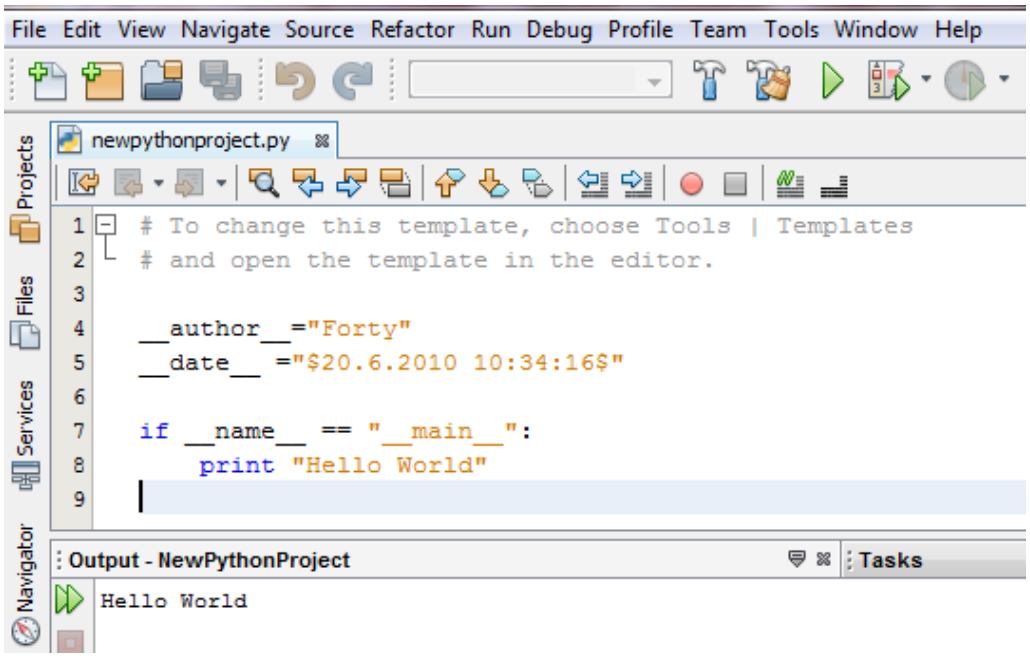
### 3.4.1 Spuštění projektu v Netbeans

Spuštění programu se provádí ikonou v menu, anebo klávesovou zkratkou F6.




**Obrázek 6: Spuštění projektu pomocí ikony**

Po kliknutí na tuto ikonu nebo stisknutí funkční klávesy F6 se spustí program a v části označované jako Output (obrázek 7) se nám zobrazí výstup programu. V našem případě se zobrazí „Hello World“.

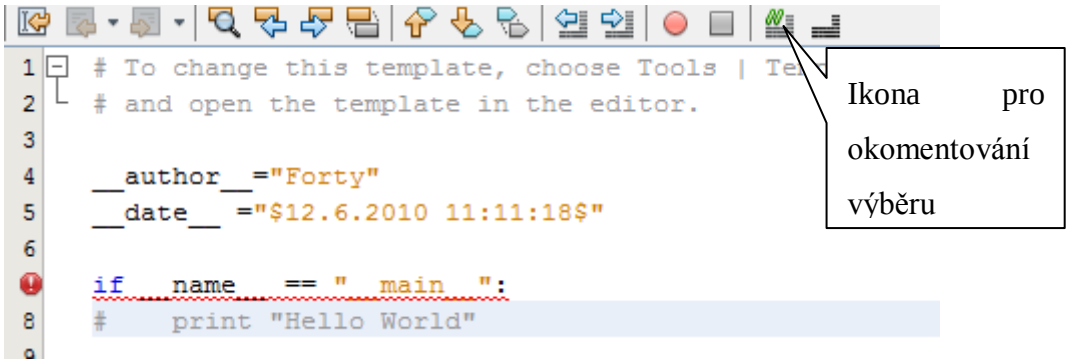



**Obrázek 7: Výstup spuštěného projektu**

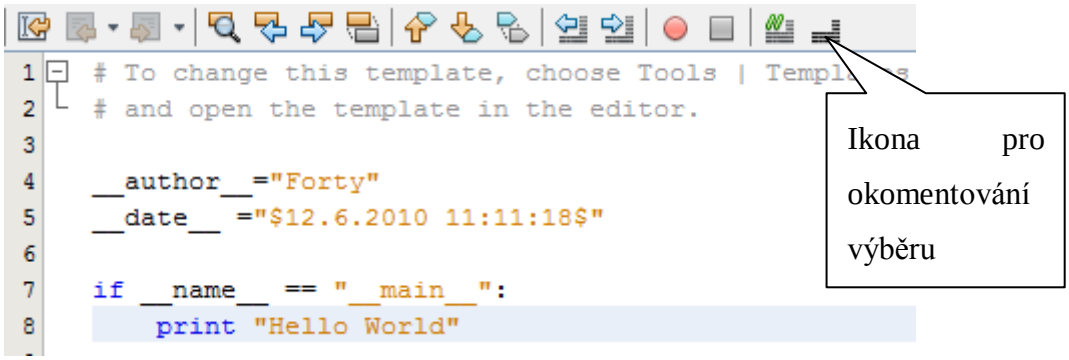
### 3.4.2 Komentář

Pro vytvoření komentáře můžeme využít ikony, která se nachází v menu (obrázek 8). Další možnost je napsaní # před řádek, který chceme okomentovat.

Pro odkomentování slouží další ikona umístněná v menu (obrázek 9). Další možností okomentování je smazání # před řádkem.



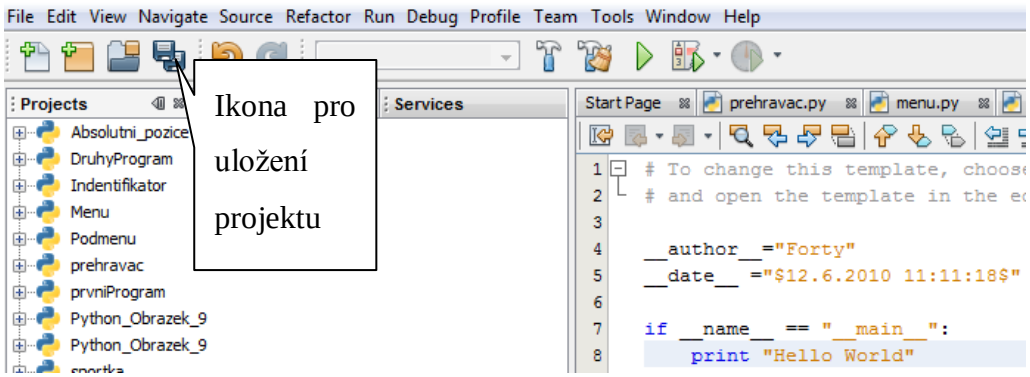

**Obrázek 8: Zakomentování řádku**




**Obrázek 9: Odkomentování řádku**

### 3.4.3 Uložení projektu

Ukládat bychom měli průběžně během psaní programu. Pro uložení můžeme použít klávesovou zkratku CTRL + SHIFT + S nebo uložíme projekt pomocí ikony v menu (obrázek 10). Pomocí této zkratky nebo ikony uložíme celý projekt. Chceme-li uložit jenom editovanou stránku, použijeme zkratku CTRL + S.




**Obrázek 10: Uložení projektu**

# 4 Trocha teorie

## 4.1 Co to je příkaz

Příkazy jsou ukončovány znakem konce řádku a blok příkazů je vytvořen odsazením.

Příkaz je instrukce, která se může provést. Pokud napíšeme příkaz do interpreta, příkaz se zpracuje a zobrazí výsledek.

## 4.2 Proměnná

Data jsou uložena v paměti počítače. K těmto datům přistupujeme pomocí proměnných. Vytváříme instance datových typů a ty následně přiřazujeme do proměnných. Proměnná představuje referenci na určitou oblast v paměti.

Proměnné v Pythonu jsou dynamicky typové. To znamená, že proměnným můžeme přiřadit různé datové typy. Proměnné se nedeklarují. K vytvoření dojde tehdy, když je jim přiřazena hodnota.

### 4.2.1 Identifikátor

Pravidla pro vytváření identifikátoru:

1) Identifikátor se může skládat pouze z velkých a malých písmen abecedy bez diakritiky, čísel a podtržítek.

2) První znak identifikátoru nesmí být číslo.

3) Identifikátor nesmí být stejný jako klíčová slova Pythonu.

Pravidla pro vytváření identifikátoru jsou povinná. Dojde – li k porušení těchto pravidel, nastane chyba programu.

### 4.2.2 Příkaz přiřazení

Příkazem přiřazení vytváříme nové proměnné a těmto vytvořeným proměnným přiřazujeme hodnoty.

Pro příkaz přiřazení používáme operátor přiřazení. Operátor přiřazení by neměl být zaměňován s rovnítkem. Operátor přiřazení přiřazuje jméno proměnné na levé straně s hodnotou na pravé straně.

```
>>> pocet = 15
>>> pi = 3.14
>>> zprava = ‘Ahoj Pythone’
```

## 4.3 Vestavěné funkce

Vestavěné funkce jsou vytvořené funkce, které jsou připravené k použití po instalaci Pythonu. Vestavěné funkce mají za cíl zjednodušit práci při programování. Tyto funkce se používají například pro práci s čísly, s řetězci, seznamy. Více o vestavěných funkcích na stránkách: http://docs.python.org/library/functions.html

### 4.3.1 Funkce range

Potřebujeme-li iterovat prvky podle aritmetické posloupnosti, bude se nám hodit právě interní funkce **range**. Kde právě jako parametr funkce dosadíme číslo, které chceme rozložit aritmetickou posloupností. Tato funkce nám vrátí seznam prvků.

*Trocha teorie*

**Funkce range s jedním parametrem**

```
range(jedenParametr)
```

jedenParametr – dosadíme číslo, které chceme rozložit aritmetickou posloupností.

Příklad:

```
range(5)
```

***Výstup interpreta:***

[0, 1, 2, 3, 4]

**Funkce range s dvěma parametry**

Pokud bychom chtěli, aby začínala posloupnost od jinud než od nuly, tak vložíme druhý parametr funkce.

```
range(parametrJedna, parametrDva)
```

parametrJedna – počáteční číslo aritmetické posloupnosti. Od tohoto čísla bude začínat aritmetická posloupnost

parametrDva – tímto číslem končí aritmetická posloupnost n-1

Příklad:

```
range(6,11)
```

***Výstup interpreta:***

[6, 7, 8, 9,10]

**Funkce range se třemi parametry**

*Trocha teorie*

Další častá varianta použití této funkce je nastavení po jakých krocích má aritmetická posloupnost přirůstat. Stačí přidat oproti předchozí verzi jeden parametr navíc.

```
range (parametrJedna, parametrDva, parametrTři)
```

parametrJedna – krok aritmetické posloupnosti

parametrDva - počáteční číslo aritmetické posloupnosti. Od tohoto čísla bude začínat aritmetická posloupnost

parametrTri - tímto číslem končí aritmetická posloupnost n-1

Příklad:

```
range(2,6,11)
```

***Výstup interpreta:***

[ 6, 8, 10]

## 4.4 Čísla

V Pythonu rozlišujeme celé, dlouhé, desetinné.

| Typ proměnné | Typ čísel | Deklarace | Rozsah proměnných |
|---|---|---|---|
| int | Celé číslo | a= 5 | −32 768 do 32 767 |
| float | Číslo s plovoucí čárkou | a= 3.14 | +(-) 1.5 .10-45 až +(-) 3.4 .1038 |
| long | Dlouhé číslo | a = 0L | −2 147 483 648 do 2 147 483 647 |

*Trocha teorie*

S čísly můžeme provádět sčítání, odčítání, násobení, dělení.

| Příklad | Popis významu |
|---|---|
| M + N | Sčítání M a N |
| M - N | Odčítání N od M |
| M * N | Násobení M a N |
| M / N | Dělení jak čísel typu integer tak reálných čísel. Výsledek záleží na typu čísel M a N. Když je alespoň jedno z čísel M a N reálné, výsledek bude též reálný. |
| M % N | Modulo: nalezne zbytek po celočíselném dělení M : N |
| M**N | Umocňování: M na N-tou |

Převzato z http://www.skil.cz/python/cztutdata.html

Více o porovnávání čísel obsahuje kapitola 4.4.2

### 4.4.1 Výrazy

Výraz je konstrukce, která obsahuje operandy, operátory a kulaté závorky. Pomocí těchto kulatých závorek můžeme ovlivnit způsoby vyhodnocení

výrazu. Výsledkem vyhodnocení výrazu je vždy hodnota. Pro výrazy je typické použití operátorů +, -, * a / . Operandem může být zavolání funkce.

```
>>> 2+2
4
>>> (50-5*6)/4
5
>>> # Dělení celých čísel vrátí celé číslo:
>>> 7/3
2
>>> 7/-3
-3
```

## 4.5 Řetězce

Řetězce jsou složeny z jednotlivých částí – znaků, které pak tvoří celek. Typy, které se skládají z jednotlivých částí, se nazývají složené datové typy. Složené datové typy mají dvě výhody:

1. Lze k nim přistupovat jako k celku
2. Lze přistupovat k jejich jednotlivým částem, z nichž jsou složeny

Více informací o řetězcích na http://howto.py.cz/cap07.htm

Zde je ukázka možností zápisu řetězců v Pythonu:

```
>>> 'Bydlím ve městě'
'Bydlím ve městě'
>>> '"To je počítač," řekl.'
'"To je počítač," řekl.'
>>> "\"To je počítač, \" řekl."
'"To je počítač," řekl.'
```

*Trocha teorie*

Mnohdy potřebuje programátor řetězec, který je rozložený přes více řádků. První z nich, který se neváže pouze na řetězce, je spojení dvou po sobě jdoucích řádků znakem zpětného lomítka.

```
hello = 'Dlouhý řetězec obsahující mnoho\n\
řádek \n\
    Spojení po sobě jdoucích řádků\
 se provádí takto.'
```

Výsledek bude následující:

***Dlouhý řetězec obsahující mnoho***
***řádek***
***Spojení po sobě jdoucích řádků se provádí takto.***

### 4.5.1 Délka řetězce

Pro zjištění délky řetezce složí funkce **len**. Tato funkce nám vrátí číslo, které udává počet znaků v řetězci.

```
>>> mesto =  "Praha"
>>> len(mesto)
```

***5***

### 4.5.2 Porovnávání řetězců

Porovnávací operátory se dají také použít na řetězce. Můžeme porovnávat libovolné řetězce. Příklad porovnávání řetězců.

```
if slovo == "Praha":
  print  "Hlavni mesto Ceske republiky"
```

## 4.6 Seznamy

Seznam je uspořádaná řada hodnot s indexem. Hodnotě ze seznamu se říká položka seznamu. Položky seznamu mohou být různého datového typu. Nový seznam vytvoříme uzavřením položek do hranatých závorek.

```
Numbers = [20, 31, 42]
```

### 4.6.1 Přístup k položkám seznamu

Pro přístup k položkám seznamu se používá index seznamu. Index v každém seznamu začíná od nuly.

```
>>> print Numbers[0]
20
```

Má-li index zápornou hodnotu, nenastane chyba programu. Index se bude počítat od konce seznamu.

```
>>> print Numbers[-1]
31
```

### 4.6.2 Délka seznamu

Délku seznamu zjistíme pomocí funkce **len**. Funkce len vrátí délku seznamu (počet položek seznamu).

```
>>>len(Numbers)
3
```

### 4.6.3 Operace se seznamy

Operátor + slouží k zřetězení seznamů. Můžeme zřetězit libovolné seznamy o různých délkách.

```
>>> a = [ahoj, dobrý_den]
```

```
>>> b = [Python, Světe]

>>> c = a + b

>>> print c
```

***[ahoj, dobrý_den, Python, Světe]***

Operátor ***** slouží k opakování seznamu v zadaném počtu.

```
>>> [ahoj] * 3
```

***[ahoj, ahoj, ahoj]***

### 4.6.4 Smazání položek seznamu

Pro smazání položek sezamu se používá funkce **del.**

```
>>> pozdrav = [ahoj, dobrý_den]

>>> del pozdrav[0]
```

***[dobrý_den]***

### 4.6.5 Vyhledávání v seznamu

Pro vyhledávání v seznamu můžeme použít: 1) operátor **in**

2) vyhledávání pomocí indexu

Operátor **in** nám zjistí zda-li je položka v seznamu.

Příklad:

```
>>> pozdrav = [ahoj, dobrý_den]

>>> ahoj in pozdrav
```

***true***

*Trocha teorie*

Vyhledání pomocí indexu se zastaví na první odpovídající hodnotě v seznamu. Není-li položka nalezena nastane chyba programu.

Příklad:

```
>>> pozdrav = [ahoj, dobrý_den]
>>> pozdrav.index("ahoj")
```

***0***

### 4.6.6 Prázdný seznam

Prázdný seznam je takový seznam, který nemá žádnou položku.

Příklad:

```
prazdnySeznam = []
```

# 5 Funkce

### 5.1.1 Definice funkce

Funkce je pojmenovaná část programu, kterou lze dále zavolat v jiné části programu. V Pythonu ji definujeme klíčovým slovem **def**. Za tímto klíčovým slovem následuje jméno funkce. Dále funkce může obsahovat seznam formálních parametrů. Následuje tělo funkce, to je blok kódu (příkazů), a proto musí být odsazeno podobně jako tělo cyklu apod.[1]. Níže máme uvedeno zapsání funkce v Pythonu.

Zápis: `def nazev(parametry):`

Za dvojtečkou pokračuje tělo funkce.

***Parametry funkce***

Funkce může mít v kulatých závorkách seznam parametrů. Parametry funkce není nutno vyplňovat všechny. Záleží na konstrukci funkce a jaké možnosti nám nabízí. Nejlépe je znázorněno využití parametrů na funkci range (kapitola 4.3.1).

Ukázka definice funkce s parametry:

```
def vypis (jmeno, prijmeni, znamka):
    print  "Jméno studenta „,jmeno
    print " Prijmeni studenta", prijmeni
    print "Dostal znamku",znamka
```

# 6 Příkazy

## 6.1 Booleovský výraz

Booleovský výraz označuje takový výraz, jehož výsledkem je booleovská hodnota. Booleovská hodnota nabývá pouze dvou hodnot. Tyto hodnoty jsou **True** a **False.** Počáteční velká písmena jsou důležitá, protože true a false booleovskými hodnotami nejsou.

```
>>> 1 == 1
```

***True***

```
>>> 1 == 3
```

***False***

Operátor == (je totožné) porovná dvě hodnoty a předá booleovskou hodnotu. V tvrzení 1 je totožné s 1 jsou oba operandy shodné, takže výraz se vyhodnotí jako **True.** V tvrzení 1 je totožné s 3 nejsou oba operandy shodné a proto se výraz vyhodnotí jako **False**.

Operátor == patří mezi relační operátory. Relační operátory jsou:

| | |
|---|---|
| x != y | x není totožné y |
| x > y | x je větší než y |
| x < y | x je menší než y |
| x >= y | x je větší nebo rovno y |
| x <= y | x je menší nebo rovno y |

Pod **x** a **y** si představujeme číselné hodnoty, řetězec.

## 6.2 Příkaz if

Příkaz **if** se skládá ze záhlaví a těla. Záhlaví začíná klíčovým slovem if, booleovským výrazem a končí dvojtečkou.

Počet příkazů v těle příkazu **if** není omezen, ale musí být alespoň jeden.

### 6.2.1 Neúplná podmínka

Booleovský výraz za příkazem if se nazývá podmínka. Je-li tato podmínka pravdivá, odsazený výraz se provede. Pokud podmínka není pravdivá, neprovede se nic.

Syntaxe je:

```
if booleovský výraz:
    příkazy
```

### 6.2.2 Úplná podmínka

Úplná podmínka je taková, kde máme dvě možnosti a podmínka určí, která z nich se provede.

Syntaxe je:

```
if booleovský výraz:
    příkazy
else:
    příkazyElse
```

### 6.2.3 Zřetězené podmínky

Pokud potřebujeme více než dvě možnosti, musíme přidat další větve. Pro přidání dalších větví použijeme sériově uspořádané podmínky.

*Příkazy*

Pro přidání další větve slouží příkaz **elif.** Počet **elif** není omezen. Pouze příkaz **else** smí být použit pouze jednou a musí být uveden jako poslední.

Syntaxe je:

```
if booleovský výraz1:
    příkaz1
elif booleovský výraz2:
    příkaz2
elif booleovský výraz3:
    příkaz3
...
...
else:
    příkazElse
```

Následující program vynásobí dvě vložená čísla. Výsledek porovná a následně vypíše, jestli jsou čísla po vynásobení kladná, rovnají se nule a nebo záporná.

```
 cislo_1 = int(raw_input("Napište celé první číslo: "))
 cislo_2 = int(raw_input("Napište celé druhé číslo: "))´
vysledek =cislo_1 *cislo_2
if vysledek < 0:
   print 'Výsledek je záporný.'
elif vysledek == 0:
  print 'Vysledek je 0'
elif vysledek > 0:
 print 'Vysledek je větší jak 0. '
```

## 6.3 Příkaz for

Příkaz **for** umožňuje iterovat prvky seznamů a řetězců (sekvence). Za řídící proměnnou se postupně dosazují všechny prvky sekvence v tom pořadí, v jakém jsou v sekvenci uloženy. K ukončení cyklu dojde po vyčerpání všech prvků seznamu, nebo všech prvků řetězce. Během cyklu není bezpečné modifikovat řídící sekvenci. Může dojít buď k vynechání, nebo opakování prvku (to platí pouze pro proměnné sekvenční typy, u neměnných to nelze již z jejich principu). Potřebujeme-li přesto změnit pořadí prvků v seznamu, přes který iterujeme, musíme iterovat přes prvky kopie původního seznamu. Kopii seznamu snadno a rychle vytvoříme pomocí subsekvence s vynechanými indexy.[3]

```
for řídícíProměnná in sekvencePrvků
        příkaz
```

Následující příklad znázorňuje použití for cyklu. Tento příklad zjistí počet znaků v řetězci.

```
retezec = 'Python'
>>> for x in retezec:
        print x, len(x)
```

***Výstup:***

Python 6

## 6.4 Příkaz while

Příkaz while slouží k opakovanému provádění příkazů. V příkazu while se nachází booleovský výraz. Na základě tohoto booleovského výrazu se vyhodnotí, zda se daný příkaz provede nebo nikoliv.

```
while    booleovský výraz
            příkaz
```

*Příkazy*

Funkce příkazu while:

1. Vyhodnocení podmínky s výstupem False nebo True.
2. Je-li vyhodnocení False, pokračuj v dalším příkazu.
3. Je-li vyhodnocení True, proveď všechny příkazy v těle a vrať se ke kroku 1.

   Tělo tvoří všechny příkazy pod záhlavím se stejným odsazením.

   Tento typ toku programu se nazývá smyčka, protože se ve třetím kroku vracíme k začátku. Pokud se podmínka cyklu nestane nepravdivou, vznikne neuzavřený cyklus a příkazy se opakují stále.

```
>>> while n=0:
        print n
        n = n-1
```

## 6.5 Příkaz break a continue

Příkaz **break** slouží k aktuálnímu ukončení cyklu for nebo while a vrátí řízení za tento příkaz. Více uvidíme na ukázkovém programu.

Příkaz **continue** způsobí přeskočení na další iteraci, kde se znovu porovná řídící podmínka v případě cyklu **while.** Do cyklu **For** se dosadí další člen příslušné podmínky.

Ještě za zmínku stojí to, že můžeme použít větev **else**. Tato větev se provede po regulérním ukončení cyklu.

```
for x in range(2, 10):
          delitel=2
          if x % delitel == 0:
            print x, 'lze vydelit', delitel
            break
          else:
            print x, 'nelze delit'
print'Ukonceno'
```

Tento program vypočítá z definovaného rozsahu, jestli je číslo dělitelné dělitelem. V našem případě je číslo dělitelné dvěma. For cyklus s pomocí funkce range vezme nejdříve číslo dvě. Číslo dvě je vyděleno a porovná se v podmínce if . Nastane-li rovnost s nulou, vytiskne se číslo, které je dělitelné dělitelem. Následuje klíčové slovo break. Toto slovo dokáže to, že se cyklus for ukončí a řízení bude pokračovat až za tímto cyklem. To znamená, že řízení přeskočí na `print 'Ukonceno'`.

Program naleznete na CD ve složce příklady/dělitel

Výsledek programu bude vypadat takto:

***2 lze vydelit 2***

***Ukonceno***

Nyní ukázka stejného programu, ale s použitím **continue.**

```
for x in range(2, 10):
          delitel=2
          if x % delitel == 0:
             print x, 'lze vydelit', delitel
             continue
          else:
             print x, 'nelze delit'
print'Ukonceno'
```

For cyklus s pomocí funkce range vezme nejdříve číslo dvě. Číslo dvě je vyděleno a porovná se v podmínce If . Nastane-li rovnost s nulou, vytiskne se „číslo je dělitelné dělitelem“. Následuje klíčové slovo continue. Toto slovo zajistí, že zbytek těla programu v cyklu bude ignorováno a dojde znovu k porovnání podmínky cyklu for. To znamená, že řízení přeskočí na `for x in range(2,10):` Následně se zvýší číslo o jedničku a následuje celý cyklus opět až ke klíčovému slovíčku continue a celé se to znovu opakuje, dokud platí podmínka for cyklu.

Výsledek programu s použitím klíčového slova continue bude vypadat takto:

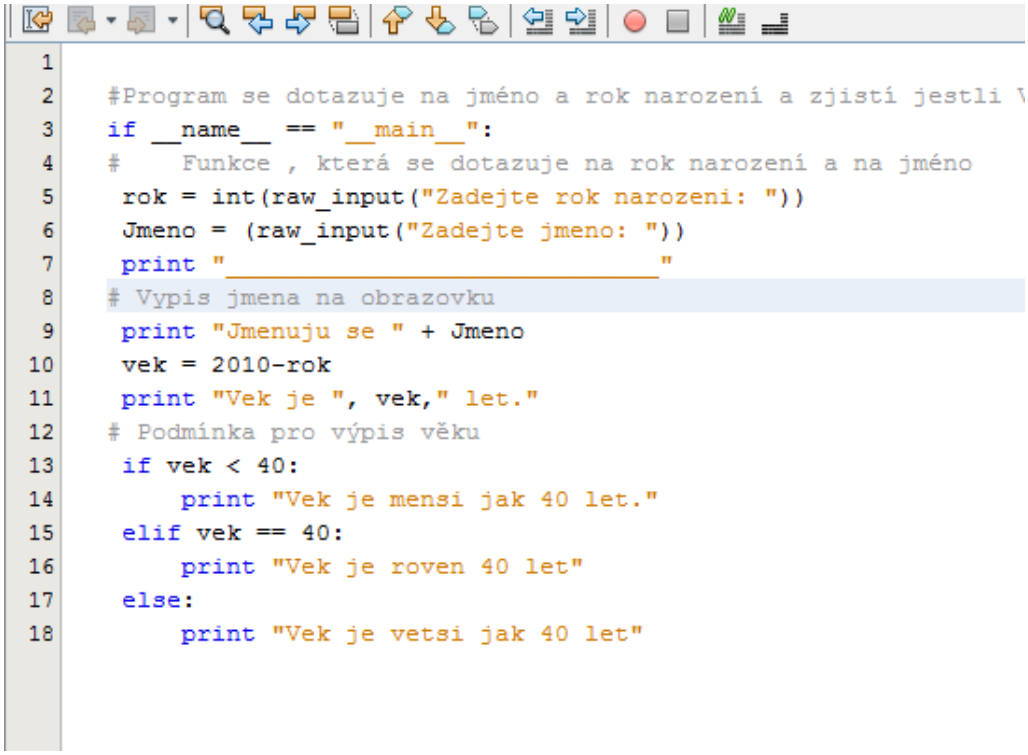
***2 lze vydelit 2***
***3 nelze delit***
***4 lze vydelit 2***
***5 nelze delit***
***6 lze vydelit 2***
***7 nelze delit***
***8 lze vydelit 2***
***9 nelze delit***
***Ukonceno***

**Cvičení:**

1. Vytvořte program, který vypíše na obrazovku všechna prvočísla v rozsahu od 2 do 20.
2. Vyzkoušejte si násobení, dělení a další matematické operace pomocí interpretu.

# 7 První program

První program je jednoduchý. Program nám bude vypisovat jméno, náš věk, dále nám zajistí, aby věk nebyl vyšší než čtyřicet let. Pro data jako je jméno a rok narození se program dotáže. Program naleznete na CD ve složce příklady/prvníProgram.



```
1
2   #Program se dotazuje na jméno a rok narození a zjistí jestli V
3   if __name__ == "__main__":
4   #    Funkce , která se dotazuje na rok narození a na jméno
5    rok = int(raw_input("Zadejte rok narozeni: "))
6    Jmeno = (raw_input("Zadejte jmeno: "))
7    print "_____"
8   # Vypis jmena na obrazovku
9    print "Jmenuju se " + Jmeno
10   vek = 2010-rok
11   print "Vek je ", vek," let."
12  # Podmínka pro výpis věku
13   if vek < 40:
14       print "Vek je mensi jak 40 let."
15   elif vek == 40:
16       print "Vek je roven 40 let"
17   else:
18       print "Vek je vetsi jak 40 let"
```

**Obrázek 11: První program - jak by měl vypadat**

`rok = int(raw_input("Zadejte rok narozeni: "))` – funkce raw_input zajistí, že se dotáže pomocí parametru, který je uveden v závorkách. Po zadání hodnoty se nám hodnota, kterou jsme zadali, přiřadí pod proměnou rok.

`print "Jmenuju se " + Jmeno` – tato část nám „vytiskne“ na obrazovku to, co je pro nás důležité.

# 8 Moduly

V této kapitole se zaměříme na moduly. Co to jsou moduly, k čemu nám slouží, přehled nejzákladnějších modulů a ukážeme si, jak modul vytvořit.

## 8.1 Co to jsou moduly

Modul slouží k uspořádání velkých projektů. Pomocí modulů se snáze pracuje s projekty.

Modul je samostatný soubor obsahující kód v programovacím jazyce Python. Modul obsahuje skupinu funkcí a proměnných.

## 8.2 Příkaz import

Jestliže chceme využívat funkce z nějakého modulu, tak v programu uvedeme příkaz **import**. Existují tři možnosti syntaktického napsání příkazu **import**.

Ukázka celého importu je následující:

```
import math
```

Tento příkaz vyhledá modul Pythonu, podle daného jména. Zanalyzuje obsah modulu a zpřístupní jej.

Ukázka částečného importu z modulu.

```
from názevModulu import název1,název2 ,…..
```

Každý objekt s názvem název1, název2, z modulu názevModulu je přístupný importujícímu kódu.

Ukázka importu všeho co modul nabízí.

```
from názevModulu import *
```

Znak * zastupuje všechna jména všech funkcí, která jsou v modulu obsažena

## 8.3 Vytvoření vlastního modulu

Pokud budeme chtít nějakou část programu často využívat, můžeme si vytvořit modul, který jenom importujeme a použijeme. Vytvoření modulu není nic složitého. Jde jenom o jednoduchou úpravu programu. Mělo by platit, že v modulu by mělo být vše ve funkcích. Například definujeme si funkci pozdrav.

```
def  pozdrav():
       return „Zdravím všechny z mého modulu!“
```

Tento kód uložíme pod názvem *pozdravy.py* . Následně vytvoříme nový projekt a uložíme ho na stejné místo jako *pozdravy.py.* Do prázdného projektu napíšeme:

```
import pozdravy
     print pozdravy.pozdrav()
```

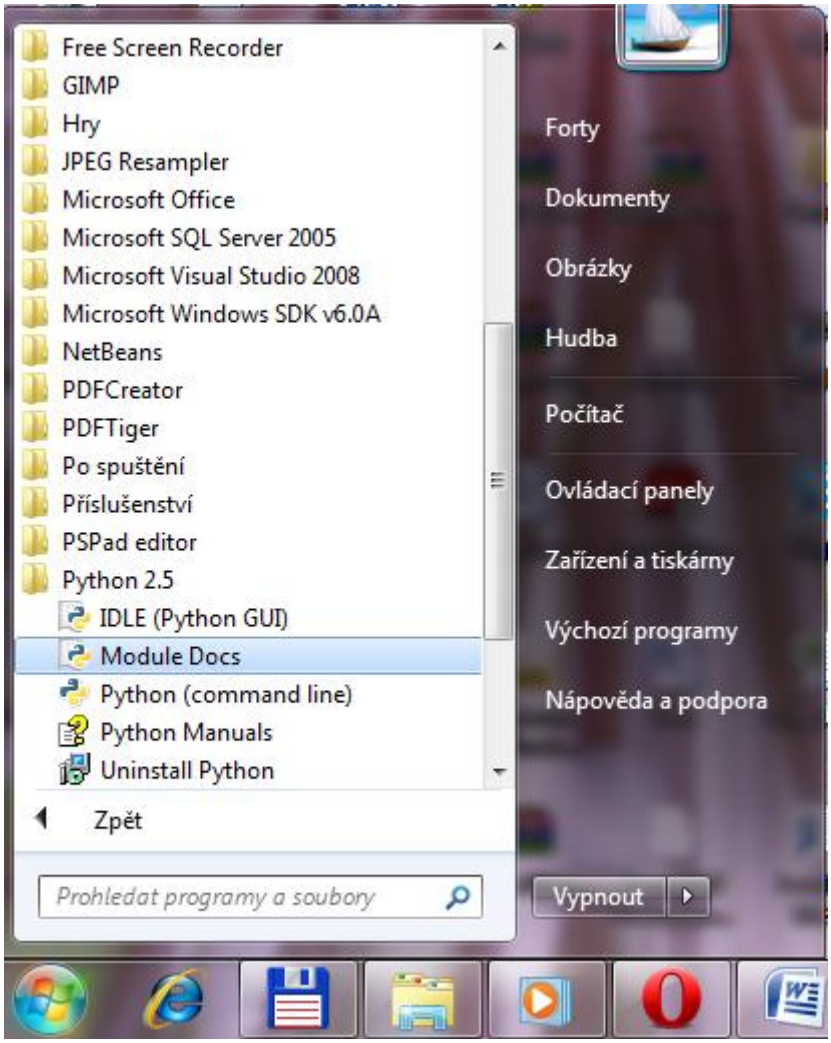
Program vypíše:

***Zdravím všechny z mého modulu!***

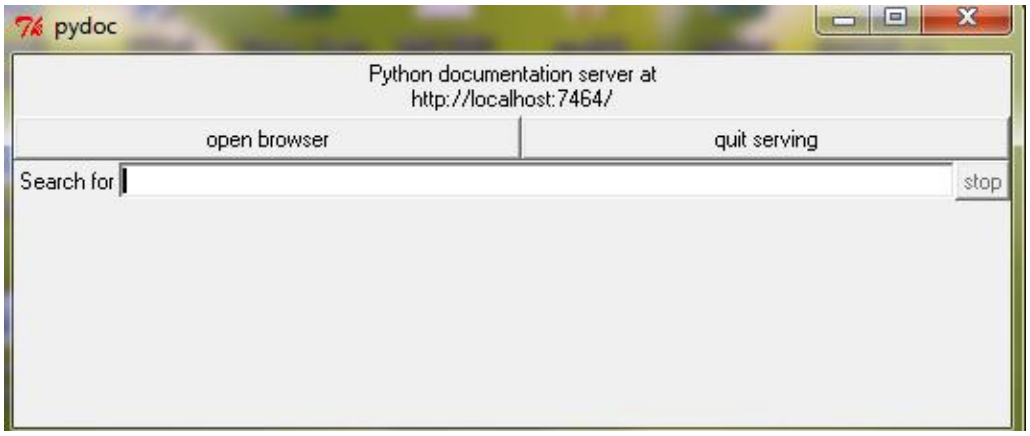
To je zatím vše, co se týče problematiky vytváření vlastních modulů.

## 8.4 Základní moduly

Distribuce Python obsahuje mnoho modulů, které je možno stáhnout a používat. S tímto tvůrci počítali, a proto vytvořili modul docs, který je přímo součástí Pythonu. Tato aplikace vytváří přehled modulů, které jsou dostupné v distribuci. Aplikaci Module Docs můžeme spustit pomocí menu Start (Obrázek 12).

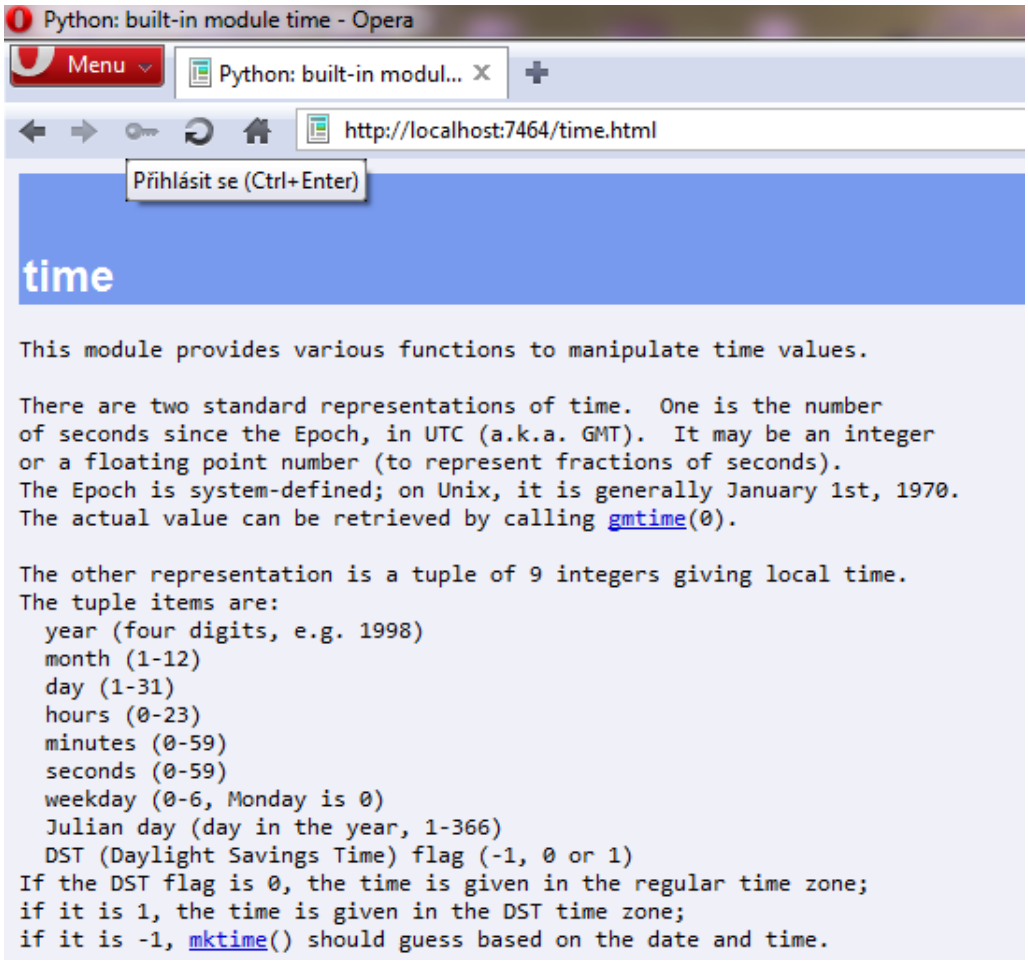



**Obrázek 12: Aplikace Module Docs v menu Start**




**Obrázek 13: Spuštěná aplikace Module Docs**

Po spuštění programu (Obrázek 13) si hned můžeme napsat, co bychom potřebovali vědět. Například napíšeme „time“ a program nám najde, v jakém modulu se nachází time a spustí Váš prohlížeč. Do prohlížeče se načte dokumentace vygenerovaná modulem pydoc (Obrázek 14)




**Obrázek 14: Spuštění internetového prohlížeče**

Pokud si chceme vytvořit náhled všech dostupných modulů, můžeme použít Pythonskou dokumentaci. V této dokumentaci je většina modulů popsána a dokonce je uvedeno i několik příkladů použití.

Velice zajímavá funkce, která se často využívá je funkce dir. Tato funkce nám vypíše z každého modulu funkce a proměnné, který daný modul obsahuje.

**Cvičení:**

1. Vytvořte vlastní modul, který bude vypisovat prvočísla od 2 do 30.

2. Vytvořte funkci, která bude vytvářet odmocninu. Výsledek odmocniny bude vypsán na obrazovku. Pro výpočet odmocniny bude využito modulu **Math.**

# 9 Objektově orientované programování

Pro dostatečné pochopení problematiky doporučuju prostudovat knihu Object-Oriented Programming in Python [32]. V této knize naleznete všechny informace týkající se objektově orientovaného programování.

V následujících podkapitolách naleznete základní rysy objektově orientovaného programování. Jedná se pouze o popis konstrukcí, které jsou v jazyce Python k dispozici. Není to tedy žádné pojednání o objektovém programování jako takovém, protože na to v této práci nebyl dostatečný prostor.

## 9.1 Základní pojmy objektově orientovaného programování

**Objekt –** vychází z reálného světa. Má dva charakteristické rysy.

- **Všechny objekty mají stav**
- **Všechny objekty mají chování**

Objekt má své vlastnosti a může vykonávat určité činnosti. Každý objekt je instancí třídy objektů. Třída objektu definuje množinu činností, které mají všechny objekty této třídy společné. Jednotlivé objekty stejné třídy se od sebe tedy liší pouze hodnotami vlastností.

**Třída -** Třída je základní konstrukční prvek objektově orientovaného programování. Třída slouží k vytváření objektů. Pomocí třídy definujeme vlastnosti a metody objektu.

**Atribut** – určuje vlastnost objektu.

**Metoda** – je funkce přidružená k určité třídě.

Python je objektově orientovaný jazyk. V této kapitole se budeme zabývat:

a) Dědičností

b) Vícenásobnou dědičností

c) Polymorfismem

d) Zapouzdřením

Objekty nejsou jenom instance třídy, ale i samotná třída je objekt v Pythonu. Mezi objekty v Pythonu patří moduly, uživatelské funkce.

## 9.2 Definování třídy

Třída se v Pythonu definuje pomocí **class**.

Vzor:

```
class VytvořenáTřída
        Tělo
```

`VytvořenáTřída` je jméno třídy.

`Tělo` je seznam příkazů jazyka Python. Může se jednat o přiřazení proměnných, definice funkcí.

Popis této konstrukce si lépe představíme na následujícím příkladu. V příkladu je vytvořena třída auto. V této třídě je definována funkce ukazZnacku, která slouží výpisu značky vozu.

```
class auto:
          def __init__(self, z = 'Fiat'):
          self.znacka = z
        def ukazZnacku(self):
          print 'Toto je auto:', self
          print 'Znacka:', self.znacka
```

## 9.3 Dědičnost

Dědičnost je často využívaná kvůli zjednodušení a lepší orientaci v kódu. Vysvětlil bych to opět na příkladu, kde využijeme předcházející příklad

(kapitola 9.2).

Příklad popisuje vytvoření třídy barvaAuta, který využívá dědičností třídu auto z příkladu z kapitoly 9.2. V třídě barvaAuta je definována funkce ukaz, která nám vypíše na obrazovku značku auta a typ auta.

```
class barvaAuta (auto):


    def __init__(self, barva = modra):


        auto.__init__(self, znacka = 'Fiat')
         self.barva = barva
  def ukaz(self):


        auto.ukazZnacku(self)
         print 'Typ auta:', self.barva
```

```
autoDoMesta = barvaAuta()
autoDoMesta.ukaz()
```

Zavolání a výpis bude vypadat takto:

**Toto je auto: <__main__.AutoDoMesta**

**znacka: Fiat**

**barva: modra**

## 9.4 Postup Pythonu při hledání atributu

Python nejdříve projde třídu objektu. Následně pokračuje prohledáváním první rodičovské třídy. Stejný princip se aplikuje i na předky této třídy. Pokud je však atribut nenalezen, přesune se na další rodičovské třídy. Pokud nastane varianta, že interpret nenajde žádný hledaný atribut, vyvolá se výjimka.

Nastane-li situace a bude chtít volat metodu z některých rodičovských tříd, použijeme zápis podobný jako je tento: ***auto.ukazZnacku(self).***

Zde voláme funkci ukazZnacku, která se nachází ve třídě auto a je součástí objektu self.

## 9.5 Třídní a statické metody

Třídní metoda je taková, která dostane místo instance třídu samotnou. Dostane ovšem první argument. U třídní metody není důležité, jestli je volána jako metoda instance nebo funkce, která je definována ve funkci. Pokud vytvoříme metodu, která bude třídní, vytvoříme vlastně klasickou funkci. Tato funkce bude obsahovat jako první argument odkaz na třídu a poté se s funkcí **classmethod** zkonvertuje.

Statické metody jsou takové metody, které nepřibírají žádný argument, v němž je instance nebo třída. Mohli bychom si to představit jako funkce, které jsou definované ve třídě. Jejich sestrojení je obdobné jako u předchozího typu metod. Místo funkce classmethod použijeme funkci **staticmethod.**

# 10 Grafické prostředí - moduly pro použití grafického prostředí

Jak již bylo zmíněno v předchozí kapitole, Python nabízí velké množství modulů a to se týká i vývoje grafického prostředí (GUI). Mezi nejznámější patří Tkinter a mezi novější a možná i více používaný patří WxPython. Záleží již na každém, co si zvolí, nebo jaký modul se mu více bude líbit. Ukážeme si zde ukázku obou těchto modulů.

## 10.1 Tkinter

Tkinter je modul, který slouží k vytváření okenních aplikací a jejich práci s nimi. Aplikace vytvořené pomocí Tkinter vypadají stejně v Unixech, Windows tak na MAC. Skládá se z mnoha menších modulů. Popsány jsou ty nejzákladnější funkce, které se využívají právě v tomto modulu.

### 10.1.1 Self

Prvním argumentem jakékoliv metody je instance třídy, nad kterou je volána. Je zvykem jí nazývat self. Více informací o self na strankách: http://ibiblio.org/g2swap/byteofpython/read/self.html

### 10.1.2 Správce rozmístění Pack

Správce rozmístění pack, vkládá jednotlivé komponenty do řádků a sloupců. Správce můžeme ovládat parametry fill, expand, side.

- Automatické rozšiřování - pokud potřebujeme umístnit komponentu do framu, aby se vyplnil celý jeho obsah. Příklad zápisu: `button.pack()`
- Při zvětšování okna, se zvětšuje i rodičovský element – pro použití takového efektu je nutno použít parametru fill a expand. Příklad zápisu: `button.pack(fill=BOTH, expand=1)`

- Umístění komponent nad sebou - použijeme pack bez parametru.
- Umístnění komponent nad sebou se stejnou velikostí – použijeme pack s parametrem fill. Příklad zápisu: `button.pack(fill=X)`
- Umístnění stejně velkých komponent vedle sebe – použijeme parametr side a fill. Příklad použití: `button.pack(side=LEFT fill=Y)`

Podrobnější informace o správci komponent pack naleznete na stránkách: http://tkinter.programujte.com/pack.htm

### 10.1.3 Nejzákladnější funkce Tkinter

Mezi nejzákladnější třídy Tkinter patří Button, Canvas, Frame. Checkbutton, Label, Listbox, Text, Radiobutton, Message.

**Frame** – jednoduchý kontejner, který nám sdružuje všechny komponenty pohromadě

```
frame = Frame (self,uvod)
frame.pack (uvod)
```

`frame` – lokální proměnná, do které je inicializován widget Frame

`frame.pack(uvod)` – viz.kapitola 10.1.2

**Button** – nám slouží k vytvoření tlačítka, na které běžně klikáme. Dá se toho na tlačítkách mnoho nastavit a základní syntaxe je takto:

```
self.button = Button(frame, text="AHOJ", fg="blue",
command=Pozdrav)
```

Parametry funkce:

`frame` – znamená do jakého framu chceme tlačítko vložit

`text` – text, který se objeví na tlačítku

`fg` - barva, kterou bude mít písmo na tlačítku

`command` – příkaz, který se provede po stisknutí tlačítka (např. definovaná funkce, spuštění dalšího okna atd.)

**Checkbutton** – Nabývá dvou stavů (zaškrtnuté, není zaškrtnuté). Nastavená hodnota bez zaškrtnutí je standardně nastavena na 0 a pokud je zaškrtnuta - nastaveno na 1. Hodnoty (0 nebo 1) lze zaměnit a to pomocí **onvalue** a **offvalue.** Pokud je potřeba více stavů použijeme Listbox.

```
from Tkinter import *
     pokus = Tk ()

    check = Checkbutton(pokus, text="vyzkousej", fg="green")
    check.pack()
```

Parametry třídy **Checkbutton** :

`text` - text, který se objeví u checkbuttonu

`fg` - barva, kterou bude mít text

`command` - příkaz, který se provede po stisknutí tlačítka (např. definovaná funkce, spuštění dalšího okna atd.)

**Label** – slouží jako štítek pro psaní textu, nebo vkládání obrázku a nabízí nám velice mnoho možností využití a nastavení

```
  from Tkinter import *
    pokus = Tk ()

    stitek = Label (pokus, text="Toto je label.",fg="green",bg="white",
font="Arial 16")
    stitek.pack()
```

`bg` – nastavení pozadí labelu

`text` – text, který se má vypsat

`fg` – barva, kterou bude mít písmo

`font` – příkaz, který určuje, jakým stylem bude písmo zobrazeno

`padx` – horizontální mezera mezi obsahem labelu a okrajem, standardně nastaveno na 1 pixel

`pady` – vertikální mezera mezi obsahem labelu a okrajem, standardně nastaveno na 1 pixel

**Radiobutton** – slouží pro výběr z více hodnot, implementace je obdobná jako u checkboxu

```
  from Tkinter import *
    pokus = Tk()

  radio = radiobutton(pokus, text="Cislo 5", fg="green", value = 5)
  radio1 = radiobutton(pokus, text ="Cislo 2", value =2)

  radio.pack()
  radio1.pack()
```

`text` – text, který se objeví u radiobuttonu

`fg` – barva, kterou bude mít text

`value` – hodnota, kterou bude mít po klinutí na tento radiobutton

`command` – příkaz, který se provede po stisknutí tlačítka (např. definovaná funkce, spuštění dalšího okna atd.)

**Menu** – k zpřehlednění možností programu. Komponenta slouží k vytvoření menu a to i rozbalovací menu s položkami.

V následujícím příkladu je vytvořeno menu. V menu jsou dvě položky, které můžeme zvolit a základě zvolení položky se nám provede definovaná funkce.

```
from Tkinter import *
  modul = Tk()
  def pozdrav():
      print "Ahoj, vítám tě!"
  def konec():
       menuZak = Menu (pokus)
       menuZak.add_command(label="Ahoj!", command=pozdrav)
       menuZak.add_command(label="Konec", command=konec)
       modul.config(menu=menuZak)
```

Další velice často používaný způsob je vytváření podnabídky. K rodičovskému menu se připojují pomocí **add_cascade.**

Příklad je podobný jako předcházející. Rozdíl je pouze v tom, že je zde použito menu s další podnabídkou.

```
modul = Tk()
def pozdrav():
    print "Ahoj, vítám tě!"
def Cau():
    print "Zdravim, vítám tě!"

def konec():
     menuZak = Menu(pokus)
menuPozdrav = Menu(menuZak, tearoff=0)
menuPozdrav.add_command(label="Ahoj", command=pozdrav)
menuPozdrav.add_command(label="Cau", command=Cau)
menuPozdrav.add_separator()
menuPozdrav.add_command(label="Pryč", command=root.quit)
menuZak.add_cascade(label="Pozdrav", menu=menuPozdrav)
```

Menu má k dispozici mnoho funkcí, které můžeme využívat pro práci. Doporučuji přečíst referenční příručku. Uvedeny jsou jen nejzákladnější funkce pro práci s menu.

`Add` - přidáme položky do seznamu menu, mohou být i typu checkbox nebo radiobutton

`Delete (index1, index2)` – maže položky menu podle indexu. Může smazat jednu i více položek.

`index 1` – počátečný index, odkud se má začít mazat.

`index 2` – konečný index, až po tento index se vše smaže. Pokud se index 2 vynechá, smaže se jenom jedna položka s indexem 1.

**Canvas** – třída pro poskytnutí strukturované grafiky. Slouží ke kreslení grafů, kreseb, znázornění geometrických útvarů

Nejdůležitější metoda je **create.** Pomocí této metody se přidávají prvky na plátno.

Příklad vytvoří čtverec, pomocí modulu canvas.

```
from Tkinter import *

ctverec = Tk()

ctver = Canvas(ctverec, width=300, height=200)
 ctver.pack()

ctver.create_rectangle(50, 25, 150, 75, fill="green")
```

Přidávat můžeme čáry, mnohoúhelníky, čtverec, oblouk, bitmapu.

**Vytvoření čáry**

```
from Tkinter import *

cara = Tk()

car = Canvas(cara, width=300, height=200)
    car.pack()

car.create_line(O, O, 150, 75, fill="blue")
```

Chcete-li se dozvědět více o možnostech dalšího využití třídy **canvas**, přečtěte si referenční příručku.

## 10.2 WxPython

WxPython je modul, který je nutné doinstalovat. Stáhnout si ho lze z http://www.wxpython.org/download.php. Návod na instalaci a dokumentaci k WxPython naleznete na http://wxwidgets.org/docs/. Komponenty jsou podobné jako u předchozího modulu, ale vypadají trochu jinak a mají i jiné metody. Podle mého názoru je WxPython lepší, z důvodu mnohdy jednodušší syntaxe, než Tkinter. Můžou se využít oba moduly v jednom programu.

### 10.2.1 Základní komponenty wxPythonu

**wx.Button** – základní a nejjednodušší komponenta je tlačítko, má podobné funkce jako u Tkinteru. Opět po stisku tlačítka se dá zavolat funkce, otevření nového okna atd. (událost).

`GetLabel()` – získání textu z tlačítka

`SetLabel(const wxString a Label)` – nastavení textu na popisku tlačítka

`wx.EVT_BUTTON(id, func)` – událost na stisknutí tlačítka, po které se vyvolá funkce, která je definovaná

**wx.ToggleButton** - opět tlačítko jako wx.Button akorát s tím rozdílem, že toto tlačítko zůstane „zamáčknuté“ po stisku. Metody jsou stejné až na metodu události na stisk tlačítka, která se drobně liší.

`wx.EVT_TOGGLEBUTTON(id, func)` – událost stisknutí tlačítka, při které se spouští určitá definovaná funkce

**wx.BitmapButton -** tlačítko, do kterého se místo popisku vkládá obrázek. Tato komponenta může nabývat stavy: zablokovaný, normální, aktivní, zmáčknutý a vybraný.

`GetBitmapDisabled()`- když je tlačítko zablokováno, tak vrátí obrázek, který bude na tlačítku zobrazen

`GetBitmapFocus()` – je-li tlačítko vybrané, tak vrátí obrázek, který bude na tlačítku zobrazen

`GetBitmapHover()` – když je tlačítko aktivní, tak vrátí obrázek, který bude na tlačítku zobrazen

`GetBitmapSelected()` – když je tlačítko zmáčknuté, tak vrátí obrázek, který bude na tlačítku zobrazen

`SetBitmapDisabled(const wxBitmap& bitmap)` – metoda k nastavení obrázku pokud je tlačítko zablokované

`SetBitmapFocus(const wxBitmap& bitmap)` – metoda k nastavení obrázku pokud je tlačítko vybrané

`SetBitmapHover(const wxBitmap& bitmap)` – metoda k nastavení obrázku pokud je tlačítko aktivní

`SetBitmapSelected(const wxBitmap& bitmap)` – metoda k nastavení obrázku pokud je tlačítko vybrané

`wx.EVT_BUTTON(id, func)` – událost, která se vykoná po stisknutí tlačítka

**wx.StaticLine** – slouží pro vytváření jak svislých tak i vodorovných čar, které se používají jako oddělovač textu, tlačítek a ostatních komponent

**wx.StaticText** – slouží pro zobrazení textu

`GetLabel()` – vrací text, který se nachází na komponentě

`SetLabel(const wxString)` – nastavuje text, který se má vypsat

**wx.ComboBox** – komponenta, která slouží pro výběr prvků z více možností

`GetValue()` – tato metoda vrací hodnotu z textového pole

`Remove(long from, long to)` – metoda, která způsobí vymazaní textu od počátečního indexu ke konečnému indexu

`SetValue(const wxString& text)` – metoda, která nám nastavuje hodnoty do comboboxu

***wx.CheckBox*** – používá se pro zjištění stavu „ano, ne“. Nabývá tedy pouze dvou stavů

`GetValue()` – pokud je komponenta zaškrtnutá, vrátí **True.** Není zaškrtnutá metoda, vrátí **False**

`SetValue()` – metoda nastavuje stav komponenty na **True** nebo **False**

`EVT_CHECKBOX(id, func)` – událost, která se provede funkci. Pokud se stav komponenty změní

***wx.RadioButton*** – používá se tam, kde je potřeba vybrat z větší skupiny stavů

`GetValue()` – je-li tlačítko zaškrtnuté, vrátí se **True.** Pokud není zaškrtnuté, vrací se **False**

`SetValue(const bool value)` – metoda pro nastavení zaškrtnutí tlačítka. Pokud je nastaveno na **True**, tlačítko je zaškrtnuté

***wx.Gauge*** – Komponenta, která nám opticky znázorňuje průběh nějakého času, tato komponenta nám nabízí dva módy

- Konečný mód je takový, u kterého víme čas potřebný na zpracování kopírování souboru.

- Nekonečný mód – nevíme, kolik času budeme potřebovat. Komponenta nám nabízí zobrazení jak vertikálně, tak i horizontálně.

`GetRange()` – metoda vrací maximální hodnotu komponenty

`GetValue()` – metoda vrací aktuální hodnotu komponenty

`SetRange(int range)` – metoda nastaví na určitou hodnotu komponentu

`SetValue(int pos)` – metoda nastaví komponentu na pozici a zároveň nastaví mód na konečný

`Pulse()` – metoda posune náplní v komponentě a zároveň nastaví mód na nekonečný

***wx.Slider*** – komponenta, která nabízí rozhraní pro číselný výběr na ose čísel. Pohybem na ose může vybrat nějakou konkrétní číselnou hodnotu

`GetMax()` – metoda, která vrátí maximální hodnotu slideru

`GetMin()` – metoda, která vrátí minimální hodnotu slideru

`GetValue()` – metoda vrací aktuální hodnotu slideru

`SetRange(int minValue, int maxValue)` – metoda, která nastaví rozsah slideru

`SetValue(int value)` – metoda nastavuje na určitou pozici

`wx.EVT_SCROLL(func)` – událost, kde se při posunutí slideru, provede definovaná funkce

***wx.ListBox*** – listbox je seznam, který v sobě uchovává různé položky. Je možno vybrat i více položek, ale záleží jaký styl je navolen

`Append(const wxString)` – metoda na vkládání položek do seznamu (Listboxu)

`GetSelection()` – metoda pro vrácení indexu položky

`GetStringSelection()` – metoda pro vrácení popisu položky ze seznamu

`Clear()`– metoda pro vymazání všech položek ze seznamu

`wx.EVT_LISTBOX_DCLICK(id, func)` – událost, při které uživatel klikne dvakrát na určitou položku, zavolá se určitá funkce

***wx.Panel*** – základní komponenta, která slouží pro skládání dalších komponent

# 11 Grafické prostředí – události, jednoduché programy

Grafické prostředí budeme vytvářet pomocí modulu WxPython. Vybral jsem tento modul proto, že jeho metody mi přijdou jednodušší a snazší na pochopení. Navíc názvy metod u každé komponenty se nazývají prakticky stejně jako například u jazyků C# a Java.

## 11.1 Události

Událostí chápeme to, že se něco děje, nebo se něco provede na nějaký podnět. To samé platí v programovacím jazyce. Například kliknutí myší na tlačítko je událost pro program. Pokud máme zaregistrován ovladač události, může tento ovladač být provázán s funkcí. Daná funkce se následně provede.

### 11.1.1 Registrace události

Registrace události je velice jednoduchá na pochopení. Skládá se ze tří kroků:

1. Vytvoření metody, která bude volána po události
2. Registrovat událost například wx.EVT_CLOSE, wx.EVT_BUTTON.
3. Svázat událost s metodou z 1. bodu.

Názorná ukázka zaregistrování události na tlačítko:

```
button = Button(self, label = 'Registrace události')
button.Bind(wx.EVT_BUTTON, self.OnClick)
```

### 11.1.2 Identifikátory

Identifikátory komponenty jsou jedinečná čísla, která nelze zaměnit a nelze udělat duplikát. Tyto identifikátory slouží především k tomu, aby bylo možné jasné poznání o jakou komponentu se jedná a tudíž jakou událost máme předat jednotlivé komponentě. Máme několik možností vytvoření identifikátorů.

a) Python si je vytvoří sám

b) Předdefinované identifikátory

c) Přidat identifikátor dle vlastního výběru

Ukázka použití identifikátoru:

```
wx.Button(parent, -2)
wx.Button(parent, wx.ID_CANCEL)
```

Jak si můžeme všimnout identifikátor je v příkladě **-2** nebo **wx.ID_CANCEL**. To znamená, že jeho výběr je na wxPythonu. Vznikne automatický identifikátor, který bude vždy záporný. Pokud vytvoříme identifikátor podle našeho výběru, bude vždy kladný. Automatické identifikátory se používají pro komponenty, se kterými již v aplikaci nebudeme pracovat, a po celou dobu se v aplikaci nemění. Hodnotu identifikátoru můžeme zjistit jednoduše pomocí metody **GetId.** Více informací ohledně událostí na stránce: http://zetcode.com/wxpython/events/.

Ukázka kódu – použití identifikátorů:

```
  import wx

  class Identifiers(wx.Frame):
      def __init__(self, parent, id, title):
          wx.Frame.__init__(self, parent, id, title, size=(200,
150))

          panel = wx.Panel(self, -1)
          grid = wx.GridSizer(3, 2)

          grid.AddMany([(wx.Button(panel,   wx.ID_CANCEL),   0,
wx.TOP | wx.LEFT, 9),
              (wx.Button(panel, wx.ID_DELETE), 0, wx.TOP, 9),

              (wx.Button(panel, wx.ID_SAVE), 0, wx.LEFT, 9),
              (wx.Button(panel, wx.ID_EXIT)),
              (wx.Button(panel, wx.ID_STOP), 0, wx.LEFT, 9),
              (wx.Button(panel, wx.ID_NEW))])


          self.Bind(wx.EVT_BUTTON, self.OnQuit, id=wx.ID_EXIT)

          panel.SetSizer(grid)
          self.Centre()
          self.Show(True)

      def OnQuit(self, event):
          self.Close()

  app = wx.App()
  Identifiers(None, -1, '')
```

Kód převzat z [22]

**Popis programu:**

`wx.Frame.__init__(self, parent, id, title, size=(200, 150))` – metoda, která zajišťuje vytvoření framu(základního okna), který má parametry **size**(velikost okna), **title**(nastavení titulku okna)

`panel = wx.Panel(self, -1)` – vytvoření panelu, ve kterém budou umístněna tlačítka

`grid = wx.GridSizer(3, 2)` – vytvoří se grid (šachovnice), která má parametry tak, že se vytvoří 3 řádky a 2 sloupce. Do nich se pak budou vkládat jednotlivá tlačítka

```
grid.AddMany([(wx.Button(panel, wx.ID_CANCEL), 0, wx.TOP | wx.LEFT, 9),
              (wx.Button(panel, wx.ID_DELETE), 0, wx.TOP, 9),
              (wx.Button(panel, wx.ID_SAVE), 0, wx.LEFT, 9),
       (wx.Button(panel, wx.ID_EXIT)),
              (wx.Button(panel, wx.ID_STOP), 0, wx.LEFT, 9),
              (wx.Button(panel, wx.ID_NEW))])
```

– metoda, která přidává jednotlivá tlačítka do gridu

`wx.Button(panel, wx.ID_SAVE)` – vytvoření tlačítka, které má automatický identifikátor wx.ID_SAVE

`self.Bind(wx.EVT_BUTTON, self.OnQuit, id=wx.ID_EXIT)` – zaregistrování události, kde na stisk tlačítka, které má identifikátor wx.ID_EXIT bude provedena funkce OnQuit

`self.Show(True)` – metoda zajistí, že se daný frame stane viditelným

`def OnQuit(self, event):` – definování funkce OnQuit, která zajišťuje ukončení aplikace

`self.Close()` – Metoda ukončí aplikaci

Příklad po spuštění by měl vypadat takto:


**Obrázek 15: Ukázka identifikátorů Windows. Obrázek převzat z [22]**

Tady bych chtěl upozornit na jednu zajímavost, která se týká rozdílného zobrazení u Linuxu. U Linuxu se k tlačítkům, které mají jako identifikátor například wx.ID_CANCEL a další identifikátory, které jsou použity v předcházejícím příkladě, přidá ikona. Tato ikona je vytvořena zcela sama pomocí implementace v Linuxu.


**Obrázek 16: Zobrazení identifikátorů LINUX. Obrázek převzat z [22]**

## 11.2 Rozmístění komponent

Existují dva způsoby rozmístění komponent v okně. Pro začátečníky se často používá absolutní pozicování (podle souřadnic). Absolutní pozicování se používá pro začátečníky hlavně proto, že je zápis jednoduchý a lehce pochopitelný. Druhý způsob zápisu je pomocí sizerů. Existuje několik typů sizerů a patří mezi ně například wx.BoxSizer, wx.FlexGridSizer, wx.GridSizer. Tyto vyjmenované sizery si zde popíšeme.

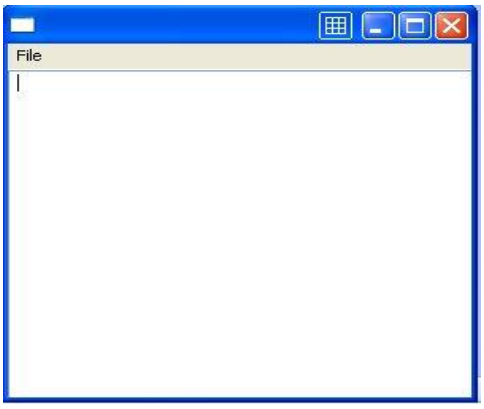
### 11.2.1 Absolutní pozicování

Absolutní pozicování je náročné z hlediska představivosti a počítání souřadnic. Tento systém pozicování, kde je nutno zadávat konkrétní souřadnice pro pozici a velikost komponenty, není vhodný pro praktické použití. Důvod, proč se neužívá v praxi je ten, že při vložení další komponenty jsme nuceni změnit souřadnice a velikost ostatních komponent v okně.

Příklad pro absolutní pozicování:

```
  import wx
  class PozicovaniAbsolutni(wx.Frame):
      def __init__(self, parent, id, title):
          wx.Frame.__init__(self, parent, id, title, size=(300,
300))
          panel = wx.Panel(self, -1)

          menubar = wx.MenuBar()
          file = wx.Menu()
          menubar.Append(file, '&File')
           self.SetMenuBar(menubar)

          wx.TextCtrl(panel, -1, pos=(-1, -1), size=(300, 250))

          self.Centre()
          self.Show(True)

  app = wx.App(0)
  PozicovaniAbsolutni(None, -1, '')
  app.MainLoop()
```

Výsledek bude vypadat takto:




**Obrázek 17: Příklad na absolutní pozicování. Příklad převzat z [23]**

`wx.TextCtrl(panel, -1, pos=(-1, -1), size=(300, 250))` – Ukázka absolutního pozicování, kde si volíme počáteční souřadnice umístnění a velikost komponenty ***(size=(300,250)).***

### 11.2.2 Umístnění komponent pomocí sizerů

Sizery mají jednu podstatnou výhodu. Spočívá v tom, že když okno zvětšíme, přizpůsobí se komponenty zvětšení okna. U absolutního pozicování nám zůstanou, i po zvětšení okna, stále stejně velké komponenty. V příkladu bude vytvořeno menu s nabídkou, které bude tvořeno pomocí automatického sizeru.

Následující příklad je stejný jako předcházející příklad, který je zobrazen na obrázku 17. V toto příkladě se používá sizer, který zajistí přizpůsobení komponent při zvětšení nebo zmenšení okna.

```
  import wx

  class AutomatickySizer(wx.Frame):
      def __init__(self, parent, id, title):
          wx.Frame.__init__(self, parent, id, title, size=(300,
300))

          menubar = wx.MenuBar()
          file = wx.Menu()
           menubar.Append(file, '&File')
          menubar.Append(edit, '&Edit')
          menubar.Append(help, '&Help')
          self.SetMenuBar(menubar)

          wx.TextCtrl(self, -1)

          self.Centre()
          self.Show(True)

  app = wx.App(0)
  AutomatickySizer(None, -1, '')
  app.MainLoop()
```

Příklad převzat z http://zetcode.com/wxpython/layout

Myslím si, že tady není co komentovat. Jenom bych dodal, že po spuštění programu vypadá výsledek stejně jako v předchozím příkladě. Rozdíl je pouze v tom, že při zvětšení okna se nám přizpůsobí i komponenty.

### 11.2.3 Sizer wx.BoxSizer

Pomocí tohoto sizeru můžeme komponenty umístnit vodorovně, nebo svisle.

Pro vytvoření komplexního layoutu můžeme vkládat jeden sizer do druhého.

Hlavní parametry a zápis vypadá takto:

```
box = wx.BoxSizer(integer orient)
box.Add(wx.Window window, integer proportion=0, integer flag
= 0, integer border = 0)
```

`box = wx.BoxSizer(integer orient)` – vytvoření BoxSizeru, kde jako parametr má orientaci BoxSizeru. Ta buď může být svislá – vertikální (wx.VERTICAL), nebo vodorovná - horizontální (wx.HORIZONTAL).

```
box.Add(wx.Window window, integer proportion=0, integer flag
= 0, integer border = 0)
```

Parametr proportion definuje poměr změny velikosti komponent.

Parametr flag zajišťuje, jak se daná komponenta bude chovat uvnitř. Jinak řečeno můžeme použít následující možnosti (wx.LEFT, wx.RIGHT, wx.BOTTOM, wx.TOP, wx.ALL).

Parametr border nastavuje mezeru mezi jednotlivými komponentami.

## 11.3 GridSizer

GridSizer je použit v předchozím příkladu v kapitole 11.1.2. GridSizer vytvoří šachovnici, kam můžeme komponenty vkládat. Mezi jednotlivými komponentami je stejná mezera a buňka pro každou komponentu je stejně velká.

```
wx.GridSizer(int rows=1, int cols=0, int vgap=0, int hgap=0)
```

Parametr rows – nastavuje počet řádků šachovnice
Parametr cols – nastavuje počet sloupců šachovnice

Parametr vgap – nastavuje vertikální mezeru mezi jednotlivými buňkami
Parametr hgap – nastavuje horizontální mezeru mezi jednotlivými buňkami

Pro další možnosti a ukázky jiných sizerů doporučuju prostudovat stránku: http://zetcode.com/wxpython/layout/.

## 11.4 Ukázka programu s jednoduchým grafickým prostředím

Program bude imitovat losování náhodných čísel. Cílem tohoto programu je ukázka absolutního pozicování, vyvolání události, základní práce s komponentami. K tomuto programu je využit modul random, pomocí kterého získáváme náhodná čísla.

```
  import wx
  import random

  class Sazka(wx.Dialog):
      def __init__(self, parent, id, title):
          wx.Dialog.__init__(self,    parent,    id,    title,
size=(300, 220))

          cisla = []
          wx.StaticText(self, -1, 'Hozena cisla', (75, 10))
          wx.Button(self, 1, 'Zavri', (50, 150))
          wx.Button(self, 2, 'Hazej', (150, 150), (110, -1))
          seznam = wx.ListBox(self, 3, (75, 30), (150, 100),
          cisla, wx.LB_SINGLE)

          self.Bind(wx.EVT_BUTTON, self.Hod, id=2)
          self.Bind(wx.EVT_BUTTON, self.Konec, id=1)

          self.Centre()
          self.ShowModal()
          self.Destroy()
          self.Show(true)



      def Hod(self, event):
           cislo = random.randrange(0, 20)
           text= str(cislo)
           seznam.Append(text)

      def Konec(self, event):
          self.Close()

   app = wx.App()
  Sazka(None, -1, 'Sazka.py')
```

`cisla = []` - deklarace proměnné. Vytvoří se prázdný seznam, který slouží pro přidávání a uchování vylosovaných čísel.

`wx.StaticText(self, -1, 'Hozena cisla', (75, 10))` – vytvoření textu jako popisu ListBoxu

`wx.Button(self, 1, 'Zavri', (50, 150))` – vytvoření jednoho z tlačítek ve Framu

```
seznam = wx.ListBox(self, 3, (75, 30), (150, 100),
    cisla, wx.LB_SINGLE)
```
– vytvoření ListBoxu s uvedenými parametry

`self.Bind(wx.EVT_BUTTON, self.Hod, id=2)` – zaregistrování události stisku tlačítka

`cislo = random.randrange(0, 20)` – metoda, která nám získává náhodné číslo za použití modulu random

`text= str(cislo)` – převod z typu int na typ string, abychom mohli zobrazit číslo a následně ho dát do seznamu

`seznam.Append(text)` – přidá text do Listboxu

**Cvičení:**

1. Vycházejte z předcházejícího programu Sazka a vytvořte možnost nového losování. Vytvořte další tlačítko, které bude provádět nové losování.

2. Upravte program Sazka tak, aby bylo možné losovat maximálně 10krát.

# 12 Grafika – základní popis, zpracování pomocí Pythonu

Grafika a její zpracování a zobrazování zažívá v poslední době velký rozmach. Stále bude toto odvětví žádané, a proto jsem ho zařadil i do výuky programování. V této kapitole si řekneme, co je to vlastně ta grafika, z čeho se skládá, jak se k ní přistupuje a konečně zpracování grafiky pomocí Pythonu.

## 12.1 Zobrazování

Do lidského oka vstupuje světlo různých intenzit a vlnových délek. Toto světlo dopadá na fotoreceptory v oku, které vlnu vyhodnotí a informaci pošlou dál do mozku. Mozek tyto informace zpracuje, vyhodnotí a my pak vidíme barvy, tvary. Již zmiňovaná vlnová délka nám určuje odstín barvy, kterou vidíme. Intenzita nám určí, jak silně ji vnímáme. Právě těchto dvou věcí je využíváno v zobrazování u počítačů. Princip zobrazování obrazu u počítačů je založen na zobrazování jednotlivých bodů (pixelů). Každý bod nabývá určité intenzity a barvy, který je dán použitým barevným systémem (viz kapitola 12.2). Tyto body jsou velmi malé, a proto se nám na obrazovce nemusí zdát zcela viditelné. U obrazovek se udává tzv. rozlišení. Rozlišení je horizontální a vertikální počet bodů (pixelů), která je daná obrazovka schopná zobrazit. Rozlišení je udáváno tedy pomocí dvou čísel (800x600, 1024x780).

## 12.2 Barevné systémy

### 12.2.1 Prostor RGB

Při vytváření obrazu jsou použity různé barvy. Tato kombinace různých barev vychází ze základních barev barevného spektra. V obrazovkách se používají tři základní barvy (složky). Tyto barvy jsou červená (red), zelená (green) a modrá (blue). Prostor RGB se jmenuje právě podle počátečních písmen anglických názvů. Hodnoty těchto barev nabývají hodnot od 0 – 255.

Barva, kterou je zastoupena hodnota 0, nemá žádnou intenzitu. Hodnota 255 znamená, že je barva nejvíce zastoupena a tudíž má největší intenzitu. Toto vyjádření znamená 3 bitové zastoupení barev (barevných složek), které je dnes nejběžnější. Pokud chceme dostat bílou barvu, musíme zobrazit červenou, zelenou a modrou v plné intenzitě. Chceme-li různé druhy odstínů šedi, musíme zvolit stejnou intenzitu u všech barevných složek. Kdybychom chtěli převést barevný obraz na obraz s různými stupni šedi, nemůžeme tento problém řešit prostým průměrem hodnot jednotlivých barevných složek. Důvod je prostý. Lidské oko vnímá intenzitu u různých barevných složek jinak, a proto se používá empirický vztah pro výpočet jasu (označováno jako I).

$$I = 0.299\ R + 0.587\ G + 0.114\ B$$

**Obrázek 18: Znázornění systému RGB. Obrázek převzat z [24]**

### 12.2.2 Prostor RGBA

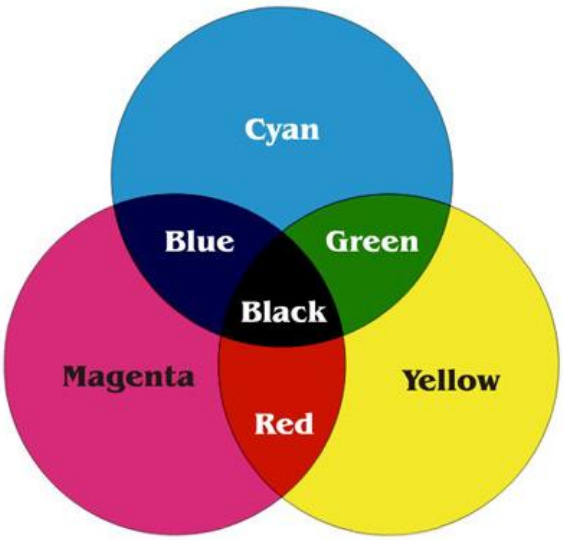
Název je podobný jako u prostoru RGB. Tento prostor má navíc mimo základních stejných barev tzv. α – kanál. Tento kanál nám určuje průhlednost jednotlivých obrazových bodů. Nejnižší číslo 0 znamená neprůhledný. Nejvyšší číslo 1 znamená plně průhledný obrazový bod. Tento prostor se

používá spíše výjimečně. Hlavní použití je, když je potřeba sloučit více obrazů do jednoho.

### 12.2.3 Prostor CMY

RGB prostor není zcela reálný z praktického života, protože je založen na aditivním skládání barevného světla. V praktickém životě je mnohem častěji používáno tzv. subtraktivní skládání světla (přidávání jednotlivých barev).

Tento prostor se skládá ze tří základních barev: tyrkysová (cyan), červenofialová (mangenta) a žlutá (yellow). Složením těchto barev nám vznikne černá barva. Takovéto skládání barev je vhodné pro tiskárny, které tisknou podle tohoto systému barev.




**Obrázek 19: Prostor CMYK. Obrázek převzat z [23]**

### 12.2.4 Prostor CMYK

Prostor CMYK vychází z prostoru CMY. Skládá se ze stejných barev jako prostor CMY. Přidané písmeno do názvu prostoru znamená, že je přidaná černá barva (BlacK). K tomuto přidání černé barvy došlo hlavně z důvodu využití systému v polygrafii. V polygrafii nestačilo zastoupení všech tří barev CMY a

výsledná „černá“ nevypadala dobře. Z toho důvodu se do prostoru přidává černá barva, která tento nedostatek řeší.

### 12.2.5 Formáty grafiky

Formáty grafiky rozdělujeme do dvou kategorií. První kategorie je vektorová grafika (grafika tvořená z rovinných útvarů). Druhá kategorie je bitmapová grafika (grafika je skládána z jednotlivých bodů). Další dělení formátů je na ztrátové a bezeztrátové (formát podporuje nějaký algoritmus pro zmenšení velikosti obrázku, aniž bychom to byli schopni postřehnout)

1. Bitmapové formáty – ztrátové
    - JPEG- určený především jako závěrečný formát u úpravy fotografií
2. Bitmapové formáty – neztrátové
    - PNG – určený na webové stránky. Byl vyvinut jako zdokonalení formátu GIF
    - GIF – komprese LZW84, možno použít i pro animace, počet barev maximálně 256
    - BMP – jednoduchý formát, nevýhoda je vysoká datová velikost obrázku
3. Bitmapové formáty – ztrátové
    - JPEG – využívá kosinové modulace, lze nastavit velikost komprimování, nejčastěji používaný formát
4. Vektorové
    - SVG – k popisování grafiky používá jazyk XML
    - VRML – používá se pro aplikace virtuální reality

## 12.3 Python a obrázky

Pro zobrazování obrázků v Pythonu není potřeba doinstalovat nějaké moduly a lze obrázek zobrazit pomocí modulu Wx. Zobrazení obrázku se vytváří pomocí metody wx.Image(self, name, index). Nejlépe tuto metodu pochopíme na následujícím příkladu. Příklad znázorňuje vytvoření grafického rozhraní pro otevírání obrázků. Obrázky se budou otevírat ve formátu jpg a gif.

```
  import os
  import wx

  SIZEW = (300,200)

  class Frame(wx.Frame):

      def __init__(self, parent, id, title):
          style=wx.DEFAULT_FRAME_STYLE ^ (wx.RESIZE_BORDER)
          wx.Frame.__init__(self,  parent,  id,  title=title,
size=SIZEW, style=style)
          self.Center()
          self.Show()
          self.panel = wx.Panel(self)
          self.panel.SetBackgroundColour('White')

          menuBar = wx.MenuBar()

          self.SetMenuBar(menuBar)
          menuFile = wx.Menu()
          menuBar.Append(menuFile, '&File')

          fileOpenMenuItem  =  menuFile.Append(-1,  '&Otevri
obrazek', 'Otevri obrazek')
```

```
    self.Bind(wx.EVT_MENU, self.Open, fileOpenMenuItem)



          exitMenuItem = menuFile.Append(-1, 'Exit', 'Konec')
          self.Bind(wx.EVT_MENU, self.Exit, exitMenuItem)
     def Open(self, event):
          filters = 'Image files (*.gif;*.jpg)|*.gif;*.jpg'
          dlg  =  wx.FileDialog(self,  message="Otevri  obrazek",
defaultDir=os.getcwd(),    defaultFile="",    wildcard=filters,
style=wx.OPEN)

         if dlg.ShowModal() == wx.ID_OK:

              filename = dlg.GetPath()
              self.SetTitle(filename)
              wx.BeginBusyCursor()
              self.image = wx.Image(filename,
wx.BITMAP_TYPE_ANY, -1)

              self.ShowP()
              wx.EndBusyCursor()

          dlg.Destroy()

      def ShowP(self):

          self.bitmap = wx.StaticBitmap(self.panel, -1,
wx.BitmapFromImage(self.image))

          self.SetClientSize(self.bitmap.GetSize())
```

```
    def Exit(self, event):

        self.Destroy()



app = wx.App()
Frame(None, -1, 'Image Viewer')
app.MainLoop()
```

`filters = 'Image files (*.gif;*.jpg)|*.gif;*.jpg'` – vytvoření proměnné filters, která slouží jako vymezení formátů

`dlg = wx.FileDialog(self, message="Otevri obrazek", defaultDir=os.getcwd(), defaultFile="", wildcard=filters, style=wx.OPEN)` – vytvoření fileDialogu, který nám umožní vybrání souboru s obrázkem

`message="Otevri obrazek"` – nastavuje název dialogového okna

`defaultFile=""` – slouží k nastavení cesty dialogového okna. Pokud je nastaveno na prázdný řetězec, je defaultně dialogové okno nastaveno na cestu, ve které je uložen projekt

`wildcard=filters` – slouží k filtrování souborů

`style=wx.OPEN` – styl dialogového okna, který je v tomto případě nastaven pro otevírání souborů a ne například pro ukládání

`filename = dlg.GetPath()` - získáme cestu k adresáři s obrázkem

`self.image = wx.Image(filename, wx.BITMAP_TYPE_ANY, -1)` – vytvoříme image, kde první parametr filename je cesta k souboru s obrázkem

`wx.BITMAP_TYPE_ANY` – slouží k tomu, že se automaticky pokusí zjistit formát obrázku

`self.bitmap = wx.StaticBitmap(self.panel, -1, wx.BitmapFromImage(self.image))` – vytvoření bitmapy, která nám daný obrázek zobrazí do panelu

**Cvičení:**

1. Modifikujte prohlížeč obrázků, který je popsán v této kapitole. Proveďte modifikaci spočívající v rozšíření prohlížení obrázků ve formátu png.
2. Prohlížeč obrázků rozšiřte o ovládání pomocí tlačítek.

# 13 Zvuk – co je to zvuk, použití v Pythonu

Produkce zvuku je dnes brána téměř jako samozřejmost. Proto je vhodné se o zvuku zmínit v následující kapitole. Kapitola bude pojednávat o vzniku zvuku, parametrech zvuku, kódování zvuku, modulech používaných pro práci se zvukem a vytvoření mp3 přehrávače.

## 13.1 Co je to zvuk

Zvuk je elektromagnetické vlnění v prostředí, které v uchu vytváří sluchový vjem. Frekvence vlnění je přibližně od 16 Hz do 20 000 Hz. Tyto hodnoty jsou jenom přibližné, protože každý člověk může vnímat jiné krajní frekvence. Zpravidla hudebníci mají větší frekvenční rozsah a daleko citlivější sluch. Pokud zdroj zvuku vydává frekvenci vlnění mimo rozsah 16 Hz – 20 000 Hz, je pro člověka neslyšitelný. Pokud je frekvence menší než 16 Hz, označuje se jako infrazvuk. Je-li frekvence větší jak 20 000 Hz, je označován jako ultrazvuk.

Zdroj zvuku může být jakékoliv těleso, které se chvěje v prostředí. Prostředí je nejčastěji vzduch, ale zvuk proniká i kapalinami a pevnými látkami. Mezi zdroje zvuku patří reproduktory, sluchátka, zvony. Lidský hlas je také chvění tělesa (hlasivek). O toto chvění se stará proudící vzduch hlasivkami. Frekvence lidského hlasu se pohybuje v rozsahu 500 – 2 000 Hz. V tomto rozsahu slyšíme nejjasněji a nejsilněji. Děj, který zahrnuje vznik zvuku a jeho šíření v prostředí, se nazývá akustika.

## 13.2 Vlastnosti zvuku

Mezi základní vlastnosti zvuku patří barva zvuku, výška a intenzita (hlasitost) zvuku. Tyto vlastnosti zvuku se v praxi často mění pomocí ekvalizátorů. Jde o nástroj, který mění výšky, tempo, hlasitost zvuku.

### 13.2.1 Barva zvuku

Zvuky mohou být ve stejné výšce, a přesto nemusejí být stejné. Mohou se lišit v barevnosti zvuku. Barva zvuku určuje počet vyšších harmonických tónů ve složeném tónu. Podle barvy zvuku rozeznáváme jednotlivé lidské hlasy, hudební nástroje.

Tóny se skládají z frekvence, která je tvořena násobky frekvence základního tónu - vyšší harmonické frekvence. Frekvenci, která má dvojnásobný počet kmitů proti kmitu základnímu, se říká druhá harmonická. V praxi to znamená, že liché násobky základního kmitočtu zvuk zostřují a místo toho sudé násobky základní harmonické frekvence zvuk zjemňují.

### 13.2.2 Výška zvuku

Výška zvuku je v přímé úměře frekvenci zvuku. Pokud je vyšší frekvence zvuku, je vyšší i výška zvuku. Pro zjištění výšky zvuku je nutné ji zjistit přístrojem a pro vizualizaci se používají indikátory vybuzení.

### 13.2.3 Intenzita a hlasitost zvuku

Hlasitost zvuku se vyjadřuje akustickým tlakem, kterým působí na sluch. S tímto souvisí i intenzita zvuku, která udává dopadající energii na plochu. Mezi těmito parametry je vazba pomocí vzorce.

**Vzorec pro výpočet intenzity zvuku**

$I = E/S*t$

E – energie, která je dána vlněním.

S – plocha na kterou energie dopadá.

T – čas po jakou dobu na ní dopadá energie.

**Vzorec pro výpočet hladiny intenzity zvuku**

$L = 10\log I/I_0$

L - hladina intenzity v decibelech (dB).

I - intenzita zvuku.

$I_0$ – intenzita prahu slyšení ( $10^{-12}Wm^{-2}$)

Práh slyšitelnosti odpovídá nejnižší intenzitě zvuku a to je přibližně 0,00001 Pa. Na druhé straně práh bolesti je 100 Pa. S větší intenzitou může dojít k trvalému poškození sluchu. Dynamika lidského sluchu je přibližně 120 dB.

## 13.3 Kódovaní zvuku

Zvuk musí být kódován do digitální podoby, aby se s ním následně nechalo pracovat v počítači. Kódování, které se používá pro zvuk je PCM modulace (pulzně kódová modulace). Při PCM modulaci se vychází ze Shannonova teorému. Podle tohoto teorému musí být použito minimálně 2krát větší frekvence při vzorkování, než je největší frekvence dosažená v signálu, který se má kódovat. Pro kódování lidského hlasu například u CD se používá frekvence 44,1 kHz. V praxi se využívá A/D převodník, který odečítá hodnoty v určitém časovém intervalu. Zpětný proces využívá D/A převodník, který digitální signál převede na analogový.

## 13.4 Formáty zvuku

Podobně jako u grafických formátů i ve zvuku jsou ztrátové a bezeztrátové formáty. Princip je opět stejný. U ztrátových formátů zvuku dojde ke ztrátě informace vlivem použitého kódování. Tato ztráta informace není podstatná, protože lidský sluch tuto ztrátu informace nepostřehne. U bezeztrátových formátů nedochází ke ztrátě informace, ale velikost souboru je o něco větší než u ztrátových formátů.

1. Zvukový formát – ztrátový

    - MP3 – ztrátový formát, který patří k jednomu z nejstarších a nejznámějších formátů. Jeho použití je univerzální a používá se skoro všude (poslech hudby, hudba k filmům). Mezi jeho výhody patří velká komprese souboru (malá velikost výsledného hudebního souboru). Nevýhoda je velká ztrátovost informace.
    - AC3 – nevýhoda toho formátu je malá komprese, takže výsledné soubory jsou dost velké. Použití tohoto formátu je u DVD.
    - WMA – formát protlačovaný firmou Microsoft. Horší výstupní parametry jako u formátu MP3. Použití tohoto formátu je minimální.
    - Vorbis – patří do kontejneru Ogg. Tento formát podporuje vícekanálový zvuk. Velice dobrá kvalita zvuku a kompresní poměr. Formát přehraje téměř, každý přehrávač.

2. Zvukový formát – bezeztrátový

- WAW – bezeztrátový formát, který vychází z PCM (pulzně kódová modulace). Z tohoto vychází i Audio CD. Nejčastěji používaný formát pro zpracování zvuku. Velikost souboru je omezena na 4 GB.
- WavPack – využívá bezeztrátovou kompresi. Komprese je velice malá (komprese 1:2). Použití především tam, kde je nutné zachovat kvalitu zvuku.

## 13.5 Python a zvuk

Pro práci se zvukem je možné použití několika modulů. Zpravidla nabízejí tyto moduly téměř shodné funkce (např. přehrávání waw souborů, nastavení hlasitosti atd.). Z hlediska jednoduchosti použití metod a dobré dokumentaci na internetu, mi přišel vhodný modul Pymedia.

Modul Pymedia nabízí přehrávání mp3, wma, ogg a dalších formátů. Modul Pymedia nemusí nutně přehrávat jenom formáty zvukové, ale může přehrávat i formáty videa (viz. Kapitola 14). Na www.pymedia.org je možnost stáhnutí poslední verze Pymedia. Na stránkách je průvodce, který nám popíše, jak má probíhat nainstalování modulu a jeho použití v Pythonu. Na stránce Pymedia nechybí ani dokumentace modulu a tutoriál, ukázka přehrávače hudby a videa.

## 13.6 Mp3 přehrávač

Mp3 přehrávač je již vytvořen na přiloženém CD ve složce MP3prehravac. Přehrávač je tedy schopen přehrát mp3 soubory. Není však problém vytvořený přehrávač přeprogramovat na přehrávání například wma souborů. Vytvořený přehrávač umožňuje vybrání písně, kterou chceme přehrát. Puštění, zastavení a pauza písničky jsou řešeny přes tlačítka. Přehrávač umožňuje ztlumení zvuku a lze nastavit hlasitost zvuku pomocí slideru.

### 13.6.1 Popis funkcí Mp3 přehrávače

```
def pust(self, event):


    global hudba
    global delka_skladby
    try:
      hudba = pymedia.Player()
      hudba.start()
      hudba.startPlayback(soundfile)
      time.sleep(1)
      delka_skladby = hudba.getLength()
      slider1.SetRange(0, delka_skladby)
      self.datas()
```

`def pust(self, event):` – vytvoření funkce, která je volána po stisknutí tlačítka play

`hudba = pymedia.Player()` – do proměnné hudba přiřadíme z modulu Pymedia Player, který bude nadále nastavovat

`hudba.start()` – spuštění Playeru

`hudba.startPlayback(soundfile)` – spuštění písně, která je předána parametrem soundfile

`delka_skladby = hudba.getLength()` – přiřazení do proměnné **delka_skladby** velikost skladby v milisekundách

```
def pauza(self, event):
     try:
       stav=hudba.isPaused()
       if stav == 0:
        hudba.pausePlayback()
        stav=hudba.isPaused()

       elif stav == 1:
         hudba.unpausePlayback()
```

`stav=hudba.isPaused()` – do proměnné **stav** se přiřazuje hodnota, která nabývá hodnoty 0 (player není pozastaven), nebo hodnoty 1 (player je pozastaven)

`hudba.pausePlayback()` – metoda, která slouží pro pozastavení (pause) playeru

`hudba.unpausePlayback()` – metoda pro opětovné puštění playeru

**Cvičení:**

1. Upravte přehrávač zvuku, který je popsán v této kapitole. Úprava spočívá ve vytvoření menu, ve kterém bude výpis stavu přehrávače. (přehrávač hraje, stop přehrávače).

2. Proveďte úpravu přehrávače zvuku, který bude rozšířen o slider znázorňující průběh přehrávání písně.

# 14 Video – co je to video, použití v Pythonu

Video je fenoménem dnešní doby a setkáme se s ním prakticky každý den. Může jít o samostatný film na nějakém médiu, televizní vysílání. Z výše zmiňovaných důvodů jsem si dovolil zařadit do kurzu práci s videem a jednoduchou ukázku přehrávače videa. V kapitole je popsáno, co je to video, z čeho se skládá, formáty videa, Python a video.

## 14.1 Co je to video

Video je technologie, která zaznamenává a přehrává sérii po sobě jdoucích obrázků. K přenosu videa se používají elektrické signály. Elektrické signály v sobě zapouzdřují složku RGB. K těmto signálům patří i zvukový doprovod, který může být jednokanálový nebo vícekanálový. Pro uchování videa slouží paměťová média. Mohou to být VHS kazety, DVD nosiče, CD a dnes i blue-ray disky.

## 14.2 Vlastnosti videa

Mezi vlastnosti videa patří Frame rate, prokládání, rozlišení, poměr stran a datový tok. Tyto vlastnosti souvisí s kvalitou videa a ty zároveň souvisí s formáty, které si popíšeme v následující podkapitole.

### 14.2.1 Frame rate

Frame rate je počet snímku za jednu sekundu. Tato vlastnost videa je velmi důležitá z důvodu plynulosti videa. Pokud je malý počet snímků za sekundu (6 - 9 snímků), je obraz blikající a nevytváří plynulý pohyb. Lidské oko vnímá více jak 15 snímků za sekundu jako plynulý obraz. V praxi se používá nejčastěji 25 a 30 snímků za sekundu.

### 14.2.2 Prokládání

Video lze skládat progresivně a prokládaně. Prokládané skládání videa využívá půlsnímků, které jsou rozdělné na sudé a liché řádky. Každý půlsnímek je potom zobrazen na půlku doby zobrazení celého snímku. Progresivní skládání je tvořeno bez půlsnímků. Místo těchto půlsnímků jsou zde celé snímky, které se zobrazují na celou dobu snímku.

### 14.2.3 Datový tok

Datový tok je počet dat, která se přenesou za jednotku času. Ve videu se udává v Mbit/s. Podle datového toku je možné určit, o jakou kvalitu videa jde. Pokud je datový tok velký, je kvalita videa dobrá.

### 14.2.4 Rozlišení

U videa se dělí rozlišení na analogové a digitální. U digitálního videa se udává rozlišení v pixelech na výšku a šířku. Například 320x640.

U analogového videa je tvořeno rozlišení řádky. Počet řádků je stanoven podle normy. Například norma PAL a SECAM má počet řádků 576.

### 14.2.5 Poměr stran

Poměr stran (obrazový formát) je dán poměrem stran vodorovné a svislé čáry. Tento poměr je tedy důležitý pro spuštění videa, abychom neměli obraz malý a naopak. Poměry stran jsou 4:3 a 16:9.

## 14.3 Formáty videa

Formáty videa jsou uloženy v kontejnerech. V těchto kontejnerech může být uložen zvuk, video, nebo obojí dohromady. Kontejnery slouží k lepšímu nakládání a přenosu souborů, které uchovávají právě video, zvuk nebo zvuk s videem.

1. AVI kontejner – nejrozšířenější kontejner pro video. Podporuje mnoho kompresí videa a zvuku. Lze ho použít i pro uložení do modernějších formátu Xvid, DiX. Jeho obrovská výhoda je v kompatibilitě přehrávání pod různými operačními systémy včetně stolních přehrávačů.
2. MPEG Program Stream – kontejner, kde je použito pro video komprese MPEG - 1 a MPEG – 2. Pro audio kompresi je využito AC3 nebo MP2. Videa s podporou tohoto formátu mají zpravidla koncovku MPG,VOB.
3. MPEG Transport Stream – použití pro digitální vysílání, AVCHD kamery a BLUE – ray přehrávače. Komprese zvuku využívá AC3 a video kompresi MPEG – 2 a MPEG – 4.
4. MP4 – oblíbený kontejner, který využívá standart MPEG – 4. Podporuje i jiné komprese například MPEG – 1 a MPEG – 2. Využití je především v mobilních zařízeních, kde je použit jako formát pro přehrávání videa. Formát pro mobilní zařízení má koncovku 3GP.
5. Matroška – nejuniverzálnější kontejner, který pojme obrovské množství různých kompresí zvuku, videa. Největší výhodou je

použití pro každého zdarma. To je jeden důvod z mnoha, proč se postupně zvyšuje použití tohoto formátu. Pro video soubory je přípona MKV.

## 14.4 Přehrávač videa

V této podkapitole využijeme již vytvořený program z www. pymedia.org. Program je dobře srozumitelný, přehledně strukturovaný a nechybí v něm základní popis funkcí. Další důvod využití již vytvořeného programu z Pymedia je velikost kódu programu. Nebylo by dobré kopírovat dlouhý kód a následně popisovat funkce přehrávače. Toto jsou důvody mého rozhodnutí pro využití již hotového programu ze serveru Pymedia.

V podkapitole se tedy dozvíme, jak vytvořený program překopírovat do Netbeans a popis funkcí videopřehrávače, které využívají modul Pymedia.

### 14.4.1 Simple VideoPlayer for Pymedia

Program je vytvořen na serveru www.pymedia.org. V sekci tutrials/instalation. Názvem programu je **Simple VideoPlayer**. Kompletní odkaz: http://pymedia.org/tut/src/vplayer.py.html. Po kliknutí na odkaz si můžeme následující kód zkopírovat a vložit ho do Netbeans. Pokud máme nainstalovaný modul Pymedia, měl by program fungovat bez problémů.

### 14.4.2 Popis funkcí VideoPlayeru

V programu je využito několik modulů, které nejsou náplní tohoto kurzu. Jedná se především o moduly Thread, Time, Os. Tyto moduly není potřeba nutně znát pro tento program. Základní funkce programu by fungovaly i bez těchto modulů. Modul Thread a Time se využívají v tomto programu pro

pokročilejší funkce. Mezi tyto pokročilejší funkce patří zobrazování odehraného času pomocí slideru.

Základní funkce přehrávače, mezi které patří výběr videa, spouštění videa, pauza videa, stop videa, využívají právě modul Pymedia. Právě tyto funkce si popíšeme.

`self.ac= acodec.Decoder(params)` – slouží pro dekódování zvuku, který doprovází video, parametry dekódování jsou předány z parametrů funkce def initAudio( self, params )

`self.vc= pymedia.video.ext_codecs.Decoder(params)` – po spuštění videa se zavolá pomocí Pymedia externí kodek (hardwarový kodek), který se nadále použije pro dekódování videa. Pokud takový kodek nelze použít, provede se následující kód

`self.vc= vcodec.Decoder(params)` – tato část kódu využívá z Pymedia modulu softwarový kodek, který se použije pro dekódování videa

`dm= pymedia.video.muxer.Demuxer(format)` – metoda slouží pro streamování videa a jeho následné rozložení na jednotlivé snímky, parametr této metody je formát, který chceme streamovat

`if dm.streams[ vindex ][ 'type' ]== muxer.CODEC_TYPE_VIDEO:` – provede se streamování videa a porovná se typ formátu videa s formáty, které jsou k dispozici pro dekódování videa

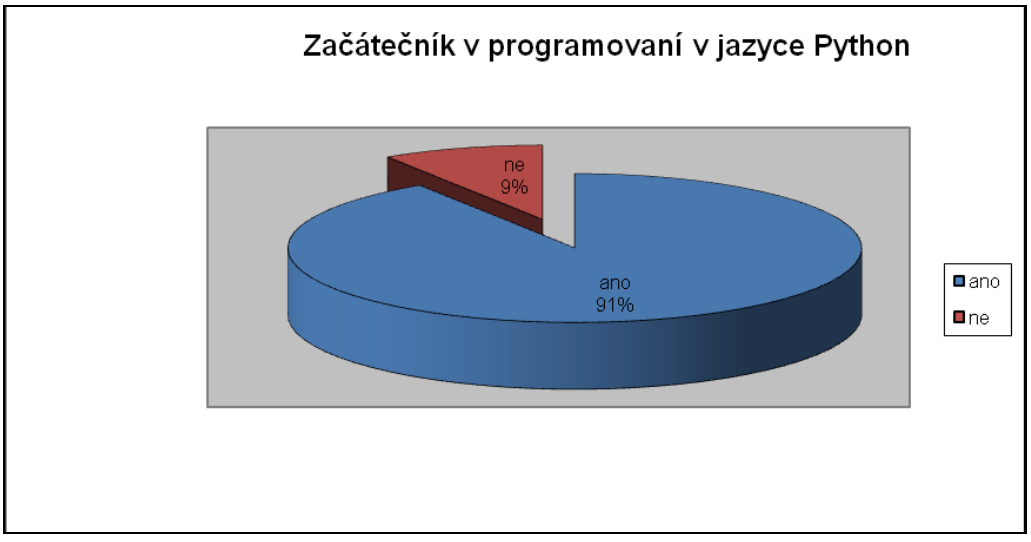
`if dm.streams[ aindex ][ 'type' ]== muxer.CODEC_TYPE_AUDIO:` – provede se streamování audia a porovná se typ formátu audia s formáty, které jsou k dispozici pro dekódování audia

# 15 Závěr

## 15.1 Výsledky práce

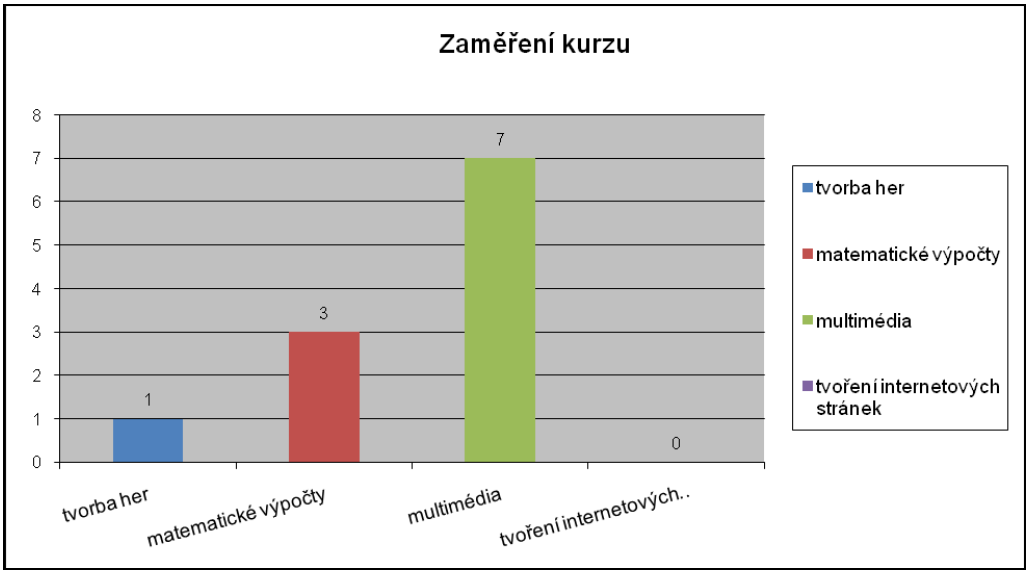
Při zpracování této bakalářské práce byly vytvořeny dva dotazníky. První dotazník byl předložen studentům předmětu KIN/PYTHA. Tento dotazník slouží k zjištění jaká metodika se má u kurzu použít a jestli je zaměření kurzu na multimedia aktuální. Dotazník byl vytvořen pomocí Google Docs a následně umístněn na systém eAMOS. Dotazník vyplnilo jedenáct studentů. Z tohoto dotazníku jasně vyplývá, že studenti jsou z velké části začátečníci v programování v jazyce Python. Dále je zcela evidentní směr kurzu. Sedm studentů z jedenácti odpovědělo, že chtějí kurz, který bude zaměřen na multimédia. V dotazníku si většina studentů přála doprovodný e-learnigový kurz. Výsledky tohoto dotazníku jsem vzal na vědomí a následně jsem vytvořil sérii výukových lekcí.

**Jste začátečník v programování v jazyce Python?**



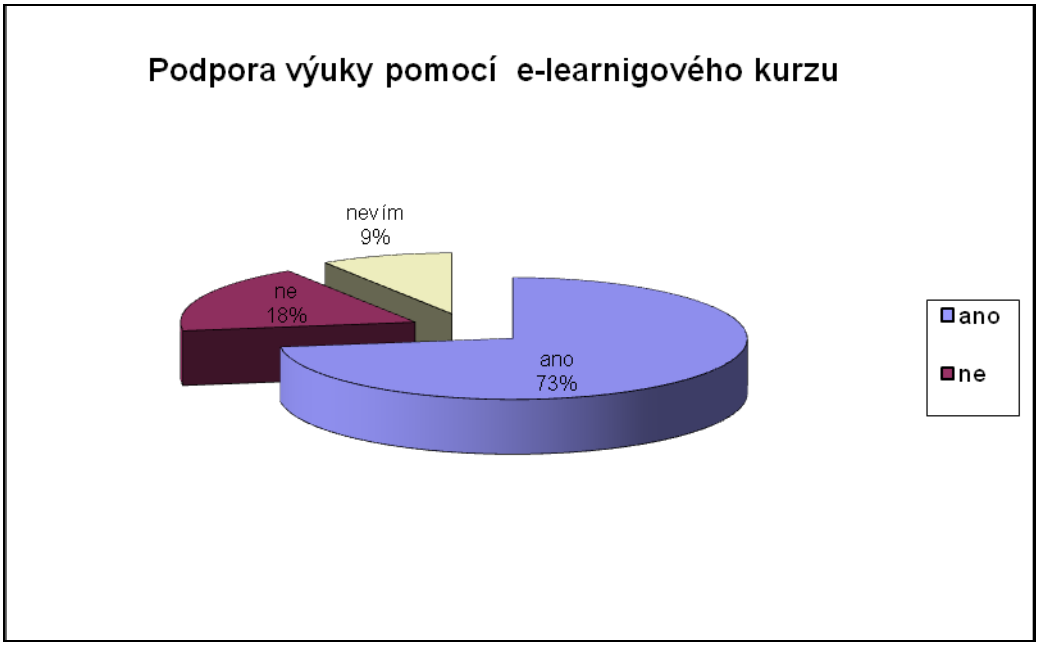

**Graf 1: Vyhodnocení úrovně znalostí programování v jazyce Python**

**Jaký směr kurzu programování v jazyce Python byste uvítal(a)?**




**Graf 2: Vyhodnocení zaměření kurzu**

**Uvítal(a) byste podporu výuky programování v jazyce Python pomocí e-learnigového kurzu ?**

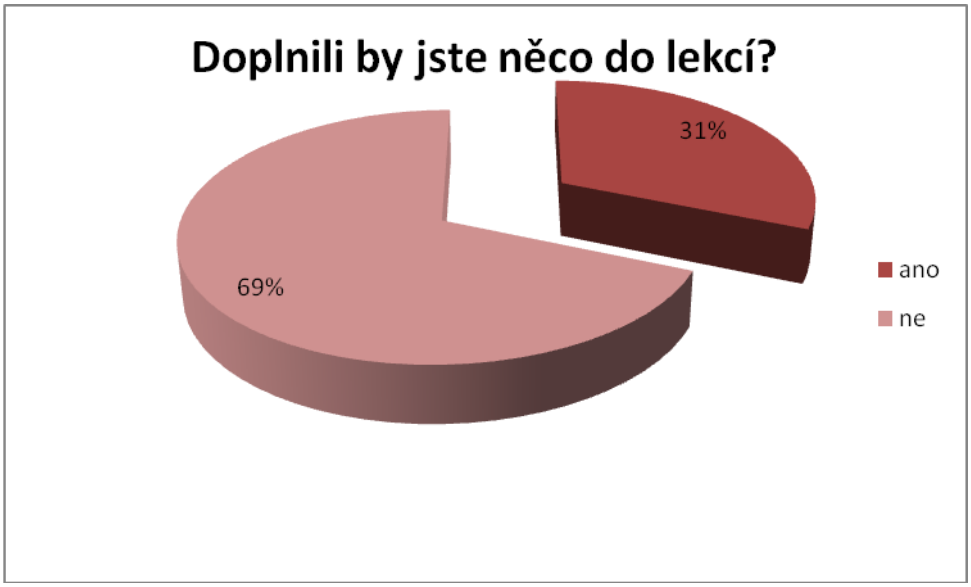



**Graf 3: Vyhodnocení podpory výuky pomocí e-learnigového kurzu**

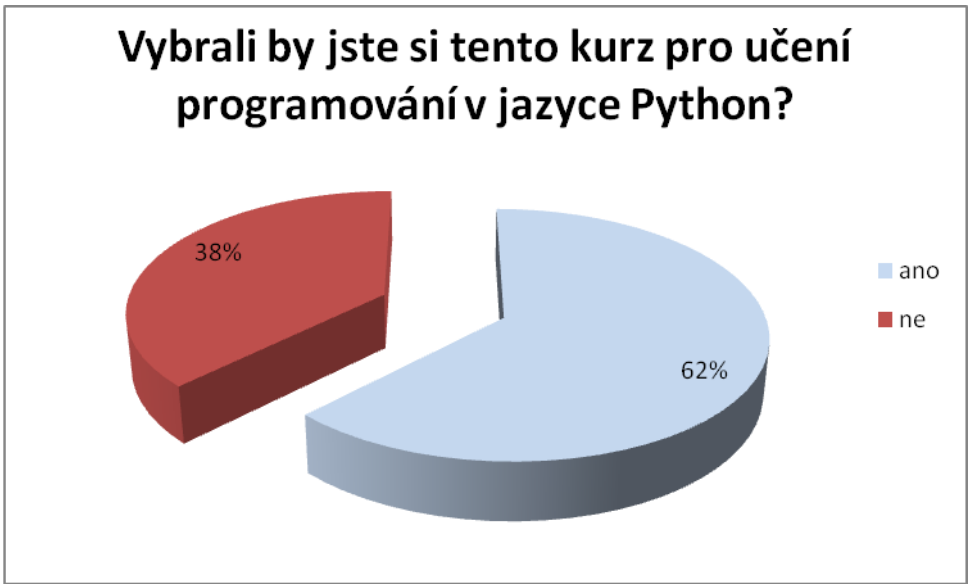
Druhý dotazník slouží k zhodnocení výukových lekcí, které byly vytvořeny pro výuku programování v jazyce Python. Dotazník byl opět podán studentům KIN/PYTHA. Dotazník je vytvořen pomocí aplikace Google Docs a následně vložen do systému eAMOS. V systému eAMOS bylo vloženo pět výukových lekcí. Na těchto pěti lekcích probíhalo zhodnocení pomocí vytvořeného dotazníku. Dotazník vyplnilo šestnáct studentů.

Dotazník se skládal ze čtyř otázek. Vyhodnocení každé otázky je znázorněno na grafech 4 -7.


**Graf 4: Vyhodnocení srozumitelnosti kurzu**




**Graf 5.Vyhodnocení úplnosti kurzu**




**Graf 6. Vyhodnocení výběru kurzu pro učení**


**Graf 7. Vyhodnocení motivace lekcí k učení**

## 15.2 Zhodnocení bakalářské práce

Cílem této bakalářské práce bylo vytvoření série výukových lekcí programování v jazyce Python pro střední školy a pro úvodní kurz programování na vysokých školách. Výukové lekce jsou směřovány k využití programovacího jazyka Python pro zpracování multimedií. Tento cíl byl splněn.

Byl vytvořen dotazník, který byl podán studentům předmětu KIN/PYTHA. Z dotazníku vyplynul zájem studentů o využití flashových animací při výuce programování. Animace se zabývají problematikou cyklů a větvení programu. Flashová animace slouží pro lepší představivost průběhu cyklu a je určena především žákům středních škol.

Dále podle dotazníku vyplynula nutnost vytvoření doprovodného

e-learnigového kurzu. Kurz slouží především jako doprovod výukových lekcí. Na e-learnigovém kurzu se nachází úkoly, které slouží pro procvičení dané problematiky. Součástí kurzu jsou e-learnigové materiály pro studenty kurzu. E-learnigový kurz je vytvořen na systému eAMOS.

Myslím, že vytvořený kurz splňuje požadavky studentů, pro které je kurz určen. Kurz obsahuje vysvětlení problematiky multimedií a současně i vysvětlení problematiky programování v jazyce Python. V práci jsou zpracovány názorné ukázky programů, které jsou současně přiloženy k práci na CD.

Část bakalářské práce byla využita v praxi. Vyzkoušení v praxi probíhalo v rámci kurzu programování v jazyce Python na Pedagogické fakultě JU. Studenti měli k dispozici pět lekcí, které si prostudovali a následně odpověděli na dotazník. V dotazníku byly čtyři otázky, týkající se pěti lekcí, které byly podrobeny výzkumu. Z dotazníku vyplynulo, že 81% studentů motivují lekce k učení. 62% studentů odpovědělo, že by si kurz vybrali k učení programování v jazyce Python. Z tohoto ověření v praxi usuzuji, že došlo ke splnění cíle bakalářské práce.

## Reference


[1] ŠVEC, Jan. Učebnice jazyka Python (aneb Létající cirkus) [online]. 16.12. 2002 [cit. 2009-10-28]. Řízení toku programu. Dostupné z WWW: <http://www.pythondocs.ic.cz/tut/node6.html#SECTION006700000000000000000>.

[2] ŠVEC, Jan. Učebnice jazyka Python (aneb Létající cirkus) [online]. 16.12. 2002 [cit. 2009-10-28]. Definice funkce. Dostupné z WWW: <http://www.pythondocs.ic.cz/tut/node6.html#SECTION006100000000000000000>.

[3] ŠVEC, Jan. Létající cirkus,Python tutoriál [online]. 13.února.2003 [cit. 2009-12-15]. Dostupný z WWW:

<http://i.iinfo.cz/r/old/data/letajici_cirkus.pdf>.

[4] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-29]. Jak se tvoří program. Dostupné z WWW: <http://howto.py.cz/cap01.htm>.

[5] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-29]. Jak se tvoří program. Dostupné z WWW: <http://howto.py.cz/cap02.htm>.

[6] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-20]. Funkce. Dostupné z WWW: <http://howto.py.cz/cap03.htm>.

[7] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-20]. Větvení a rekurze. Dostupné z WWW: <http://howto.py.cz/cap04.htm>.

[8] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-20]. Řetězce. Dostupné z WWW: <http://howto.py.cz/cap07.htm>.

[9] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-20]. Seznamy. Dostupné z WWW: <http://howto.py.cz/cap09.htm>.

[10] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-21]. Moduly a soubory. Dostupné z WWW: <http://howto.py.cz/cap10.htm>.

[11] KUBIAS, Jaroslav. Učíme se programovat v jazyce Python [online]. 2008 [cit. 2010-03-21]. Třídy a objekty. Dostupné z WWW: <http://howto.py.cz/cap13.htm>.

[12] PYTHON SOFTWARE FOUNDATION. More Control Flow Tools [online]. 1.dec.2009 [cit. 2009-12-01]. Dostupný z WWW: <http://docs.python.org/tutorial/controlflow.html>.

[13] BÍLEK, Petr. Sallyx [online]. 2005 09 11 [cit. 2009-12-01]. Python - funkce. Dostupné z WWW: <http://www.sallyx.org/sally/python/python7.php?app_invia_off=1>.

[14] BÍLEK, Petr. Sallyx [online]. 2005 09 11 [cit. 2009-12-01]. Úvod do jazyka Python. Dostupné z WWW: <http://www.sallyx.org/sally/python/python1.php?app_invia_off=1>.

[15] BÍLEK, Petr. Sallyx [online]. 2005 09 11 [cit. 2009-12-01]. Python - datové typy. Dostupné z WWW: <http://www.sallyx.org/sally/python/python3.php?app_invia_off=1>.

[16] BÍLEK, Petr. Sallyx [online]. 2005 09 11 [cit. 2009-12-01]. Python - Bloky a cykly. Dostupné z WWW: <http://www.sallyx.org/sally/python/python5.php?app_invia_off=1>.

[17] BÍLEK, Petr. Sallyx [online]. 2005 09 11 [cit. 2009-12-01]. Python - Třídy a objekty. Dostupné z WWW: <http://www.sallyx.org/sally/python/python8.php?app_invia_off=1>.

[18] BÍLEK, Petr. Sallyx [online]. 2005 09 11 [cit. 2009-12-01]. Python - moduly. Dostupné z WWW: <http://www.sallyx.org/sally/python/python11.php?app_invia_off=1>.

[19] KOSINA, Pavel . Programujte [online]. 10.06. 2005 [cit. 2009-12-02]. Python - 3. lekce. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2005060801-python-3-lekce>.

[20] KOSINA, Pavel . Programujte [online]. 24.06. 2005 [cit. 2009-12-02]. Python - 4. lekce. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=1970010105-python-4-lekce>.

[21] VOJÁČEK, Jakub. Programujte [online]. 14.11. 2008 [cit. 2010-01-02]. WxPython -základní komponenty. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2008072401-wxpython-zakladni-komponenty>.

[22] VOJÁČEK, Jakub. Programujte [online]. 08.11. 2008 [cit. 2010-01-02]. WxPython - události. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2008072800-wxpython-udalosti>.

[23] VOJÁČEK, Jakub. Programujte [online]. 03.11. 2008 [cit. 2010-01-02]. WxPython - rozmístnění komponent v okně. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2008072400-wxpython-rozmisteni-komponent-v-okne>.

[24] Regarda Design. RegardaDesign [online]. 2010 [cit. 2010-03-29]. RegardaDesign. Dostupné z WWW: <http://regardadesign.co.uk/Resources/Uploads/uploads/CMYK.jpg>.

[25] Microton s.r.o. Navajo otevřená encyklopedie [online]. 2006 [cit. 2010-03-29]. Navajo otevřená encyklopedie. Dostupné z WWW: <http://rgb.navajo.cz/rgb.png>.

[26] BODNAR, Jan. Zetcode [online]. 2007 [cit. 2009-12-02]. Layout management in wxPython. Dostupné z WWW: <http://zetcode.com/wxpython/layout/>.

[27] BODNAR, Jan. Zetcode [online]. 2007 [cit. 2009-12-02]. Events in wxPython. Dostupné z WWW: <http://zetcode.com/wxpython/events/>.

[28] JÍCHA, Vladimír. JeCh Webz [online]. 2010 [cit. 2009-12-02]. Digitální kompresní formáty. Dostupné z WWW: <http://jech.webz.cz/formaty.php>.

[29] ŽÁRA, Jiří , et al. Moderní počítačová grafika. Brno : Computer Press, 2004. 609 s. ISBN 80-251-0454-0.

[30] Video In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 25.6.2006, 9.3.2010 [cit. 2010-03-29]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Video>.

[31] GUZDIAL, Mark. Introduction to Computing and Programming in Python A Multimedia Approach. [s.l.] : Pearson Prentice Hall, 2005. 378 s.

[32] GOLDWASSER, Michael H.; LETSCHER, David . Object-Oriented Programming in Python. [s.l.] : Pearson Prentice Hall, 2008. 666 s.

[33] MACEK, Vladimír. Vyvoj softwaru, sitove sluzby, hosting, konzultace, skoleni [online]. 16. prosince 2002 [cit. 2010-04-14]. Učebnice jazyka Python (aneb Létající cirkus). Dostupné z WWW: <http://macek.sandbox.cz/texty/python-tutorial-cz/tut/node5.html>.

[34] KOSINA, Pavel . Úvod do Tkinter [online]. 2007 [cit. 2010-04-14]. Menu. Dostupné z WWW: <http://tkinter.programujte.com/menu.htm>.

[35] MACH, Petr. Wraithovy stránky [online]. 2008 [cit. 2010-03-20]. Datový model. Dostupné z WWW: < http://python.wraith.cz/zaklady-datovy_model.php>.

[36] PŘIKRYL, Petr. Jak se naučit programovat [online]. 2005/10/20 [cit. 2010-03-20]. Data, datové typy a proměnné. Dostupné z WWW: < http://www.skil.cz/python/cztutdata.html>.

[37] LUNDH, Fredrik. Effbot.org [online]. August 2006 [cit. 2010-03-20]. An Introduction to Python Lists. Dostupné z WWW: < http://www.skil.cz/python/cztutdata.html>.

[38] HAVRLANT, IT laboratoř [online]. 2009 [cit. 2010-03-20]. IT Laboratoř | Otazník v identifikátorech?. Dostupné z WWW: < http://itblob.havrlant.net/otaznik-v-identifikatorech>.

[39] KOSINA, Pavel. Úvod do Tkinter [online]. 2006 [cit. 2010-04-14]. Správce rozmístnění Pack. Dostupné z WWW: < http://tkinter.programujte.com/pack.htm >.

[40] SWAROOP C H. A Byte of Python [online]. 2003 - 2005 [cit. 2010-04-14]. The Self. Dostupné z WWW: < http://ibiblio.org/g2swap/byteofpython/read/self.html >.

[41] DAŘENA, František. PERL [online]. 2004 [cit. 2010-04-14]. Objektově orientované programování. Dostupné z WWW: < https://akela.mendelu.cz/~darena/Perl/objekty.html >.

[42] KOSINA, Pavel. Kurz Python [online]. 2006 [cit. 2010-04-14]. Python - 16. lekce. Dostupné z WWW: < http://programujte.com/?akce=clanek&cl=2006021802-python-16-lekce >.

[43] ŠVEC, Jan. Učebnice jazyka Python (aneb Létající cirkus) [online]. 16.12. 2002 [cit. 2009-10-28]. Konstrukce for. Dostupné z WWW:

<http://www.pythondocs.ic.cz/tut/node6.html#SECTION006200000000000000000>.

# Rejstřík