PYTHON

SEZNÁMENÍ S PROGRAMOVACÍM JAZYKEM

Základy Pythonu
Datové typy
Větvení

Lukáš Kotek

CO JE PYTHON ZAČ
ZÁKLADNÍ PRINCIPY PYTHONU

- **Když Bash nestačí…**
  - Python jako skriptovací jazyk
- **Ovládací utility**
  - Významné množství v linuxovým distribucích
- **Použití jako řídící vrstva**
  - Práce s knihovnami napsanými v C
- **Webové aplikace**
  - Díky frameworkům jako Django, Bottle a další
- **Pro tvorbu rozšíření**
  - Přibalen k mnoha aplikacím jako (GIMP, Cinema 4D, LibreOffice)
- **Spolupráce s .NET a Java platformou**
  - Implementace Pythonu Jython a IronPython
- **Tvorba programů s GUI**
  - Moduly pro knihovny GTK, Qt, Tk

CO JE PYTHON ZAČ
ZÁKLADNÍ PRINCIPY PYTHONU

- **Multiplatformní**
  - Standard v linuxových distribucích
  - Ostatní unixové systémy (BSD, Solaris, MacOS, AIX)
  - Podpora pod MS Windows
- **Open Source**
  - Šířen pod vlastní licencí kompatibilní s GPL[1]
- **Interpretovaný**
  - Program potřebuje k chodu program / interpreter
  - Python je jednoduchý, ale nikdy **nebude zvlášť rychlý**
- **Dynamicky typovaný**
  - Proměnná získává svůj typ až okamžikem přiřazení
- **Objektově orientovaný**
  - Objektový model založený na třídách
  - Není nutné použít, pokud to není nezbytně třeba :-)

CO JE PYTHON ZAČ
VERZE A VÝVOJOVÉ ŘADY PYTHONU

- **Python 2.6.x a starší**
  - Nepodporované zastaralé verze
  - Možno se setkat v některých OS (RHEL 6.x apod.)
- **Python 2.7.x**
  - Podporovaná do roku 2020[2]
  - Obrovské množství napsaného software
  - Původně neplánovaná verze
  - Backportuje mnoho vlastností z Pythonu řady 3
- **Python 3.x (aktuálně 3.6.x)**
  - Již od roku 2008!
  - Začištěná syntaxe jazyka
  - Mnoho nových konstrukcí
  - Řetězce defaultně používají Unicode
  - **Není** zpětně kompatibilní s Pythonem řady 2

CO JE PYTHON ZAČ
INTERPRETERY A IMPLEMENTACE PYTHONU

- **CPython**
  - Klasická a **nejčastěji používaná implementace** Pythonu
  - Napsaná v programovacím jazyce C
- **Jython**
  - Varianta Pythonu napsaná v Javě
  - Vyžaduje pro spuštění JVM
  - Přímá práce s Javovými třídami… bez použité syntaxe Javy
- **IronPython**
  - Stejný princip jako u Jythonu (i autor)
  - Napsáno v C# nad .NET
- **Pypy**
  - Implementace zaměřená na výkon
  - Obsahuje Just-in-Time compiler (JIT)

DATOVÉ TYPY

- **Základní datové typy**
  - **`int`** – celočíselný datový typ
  - **`float`** – čísla s desetinou čárkou, znak **.** jako oddělovač
  - **`str`** – řetězec, vždy v úvozovkách
  - **`bool`** – logické hodnoty `True / False`
    - Název typu je zároveň názvem funkce pro přetypování:

```
int("8")  # Vrati typ int s hodnotou 8
```

- **Práce se vstupem a výstupem**
  - **`print()`** – výpis na standardní výstup
  - **`input()`** – načtení ze standardního vstup, **vrací řetězec**
- **Užitečné funkce**
  - **`type()`** – vrací datový typ proměnné
  - **`len()`** – vrací počet prvků, např. počet znaků v řetězci

PYTHON A KÓDOVÁNÍ

- **Řetězec bytů verus unicode string**
  - Platí pro **Python 3.x**

```
u = "Jsem řetězec"      # Unicode rezetec
u.encode("utf-8")       # Po aplikaci metody

# reprezentováno jako: b"Jsem \xc5\x99et\xc4\x9bzec"
# vystupem je string bytu
```

- **Hlavička souboru se zdrojovým kódem**

```
#!/usr/bin/env python3    # Volba interperteru

# Zapis pro Python 2.x
# -*- coding: utf-8 -*-
# V Pythonu 3 je defaultnim kodovanim utf-8
```

JMENNÉ KONVENCE

- **Jak pojmenovat proměnné?**
  - Stanovuje dokument PEP8
  - Názvy proměnných jsou **case sensitive**

```
nazev_promenne   # doporuceny zapis
NazevPromenne    # vyzaduje-li to pouzivana knihovna
                 # a v oduvodnenych pripadech
```

- **Formátování výstupu pomocí metody format**
  - Doporučený způsob formátování řetězce

```
jmeno = "Lukas"
narozen = 1988
hmotnost = 95.532
retezec = "{0} narozeny {1} vazi {2:.1f} kg".format(
                  jmeno, narozen, hmotnost
          )

# Co vypise: print(retezec)?
```

VÝJIMKY

- **Struktura try – except – finally**
  - `finally` je nepovinný blok
  - Lze určit typ odchytávané výjimky

```
try:
    # Kod, který se ma vykonat (tusime potencialni chybu)
    pass    # Nedela nic, blok nesmi byt prazdny
except:
    # Pokud v bloku try dojde k vyvolani vyjimky
    # vykona se blok except
    pass
finally:
    # Vykona se vždy
    pass
```

- **Odsazení zleva je zásadní → jednotné odsazení tvoří blok kódu**

https://docs.python.org/2/tutorial/errors.html

VĚTVENÍ

- **Struktura if – elif – else**
  - Pouze příkaz **`if`** je povinný
  - Příkazů **`elif`** může být různý počet
  - Jediná konstrukce pro větvení v Pythonu
    - Python **neobsahuje** strukturu case / switch
  - Lze do sebe zanořovat

```
if VYRAZ_1:
    # Kod, který se ma vykonat, pokud je vyraz pravdivy
    pass
elif VYRAZ_2:
    # Vykona se, pokud je predchozi vyraz nepravdivy
    # a pokud je vyraz v ramci elif pravdivy
    pass
else:
    # Vykona se, pokud jsou predesle vyrazy nepravdive
    pass
```

https://docs.python.org/2/tutorial/controlflow.html#if-statements

KAM ZA PYTHONEM?
ZDROJE A VÝUKOVÉ MATERIÁLY O PYTHONU

- **Výukový kurz online**
  - Interaktivní kurz online (včetně interpreteru a úkolů)
  - https://www.codecademy.com/learn/learn-python
- **Česká komunita kolem Pythonu**
  - http://www.py.cz/FrontPage
- **Python v kostce**
  - Výborný stručný, ale výstižný zdroj v češtině
  - https://www.sallyx.org/sally/python/
- **Ponořte se do Pythonu**
  - V češtině pouze pro Python 3
  - Volně ke stažení (z edice CZ.NIC)
  - http://diveintopython3.py.cz/index.html
- **PyLadies**
  - Český kurz Pythonu
  - http://naucse.python.cz/course/pyladies
- **Oficiální web projektu**
  - Včetně špičkové dokumentace (i offline v základní instalaci)
  - http://python.org

Otázky?

ZDROJE

1) https://docs.python.org/3/license.html
2) https://pythonclock.org/
3) http://python.org
4) http://ironpython.net/
5) http://www.jython.org