# PYTHON LITE (BPYL) 1. týden

**Václav Alt**

alt.vaclav@gmail.com

vaclav-alt.github.io, heslo: `nemamradlekorici`

Konzultační hodiny: dle dohody

## Plán

1. týden: úvod, instalace
2. týden: základní orientace v syntaxi, specifika, jednodušší příklady
3. týden: datové typy
4. týden: cykly, podmínky, logické výrazy
5. týden: funkce - parametry, kontext, lambda
6. týden: test

Karel Šafr

7. týden: Moduly: decimal, numpy, statistics
8. týden: Modul pandas
9. týden: Modul matplotlib
10. týden: úvod do OOP
11. týden: úvod do OOP
12. týden: test

## Hodnocení

- 2 testy po 40 bodech
  - povinné
  - opravné testy během zkouškového
- workshop za 20 bodů

## Interpretovaný vs kompilovaný

- Kompilovaný jazyk: Program zvaný *kompilátor* přeloží zdrojový kód do strojového jazyka, výsledkem je spustitelný soubor (`.exe`, ...)/knihovna (`.so`, `.dll`, ...).
  - výhody: zachycení *compile time* chyb, obecně rychlejší
  - nevýhody: po každé změně nutno zkompilovat znovu, platform-dependent
- Interpretovaný jazyk: Program zvaný *interpret* postupuje řádek po řádku zdrojovým kódem a postupně vykonává jednotlivé instrukce.
  - výhody: interaktivní, dynamické typování, platform-indepedent
  - nevýhody: všechny chyby jsou `runtime`

**Python je interpretovaný jazyk**

Program napsaný v Python lze i *zkompilovat*, ale obnáší to komplikace - tady to dělat nebudeme.

## Vývojová prostředí

V čem psát Python? - libovolný textový editor - konzole (`ipython`, `qtconsole`, `Idle`) "live-code" - vývojové prostředí a.k.a. *integrated development environment (IDE)* - PyCharm - Spyder - hybrid - live-code konzole + prezentace (Jupyter Notebook, Jupyter Lab) - online interpret

## Kde hledat informace

- studijní opory: Bookkit
- dokumentace: https://docs.python.org/3/

## Instalace interpreta

Základní informace k instalaci - v Plus4U

**Upozornění:** V celém kurzu předpokládáme **Python 3**

Instalaci ověříme tak, že zkusíme z příkazové řádky interpreta spustit.

### Windows

1. Anaconda - Interpret a správce balíčků v jednom:
   - https://www.anaconda.com/products/individual
   - Návod na instalaci https://docs.anaconda.com/anaconda/install/windows/, obsahuje i návod na instalaci balíčků
2. Prostě Python

### Linux

1. Anaconda (osobně nedoporučuji, ale jiní lidé by mohli) https://docs.anaconda.com/anaconda/install/linux
2. Využít systémové instalace + `pip` jako správce balíčků

```
# instalace interpreta
sudo apt-get install python3
# instalace baliku numpy
pip3 install numpy
```

### MacOS

Obecně podobné jako v Linuxu, nemám zkušenost. 1. Anaconda https://docs.anaconda.com/anaconda/install/mac-os/ 2. Bez Anacondy https://installpython3.com/mac/

---

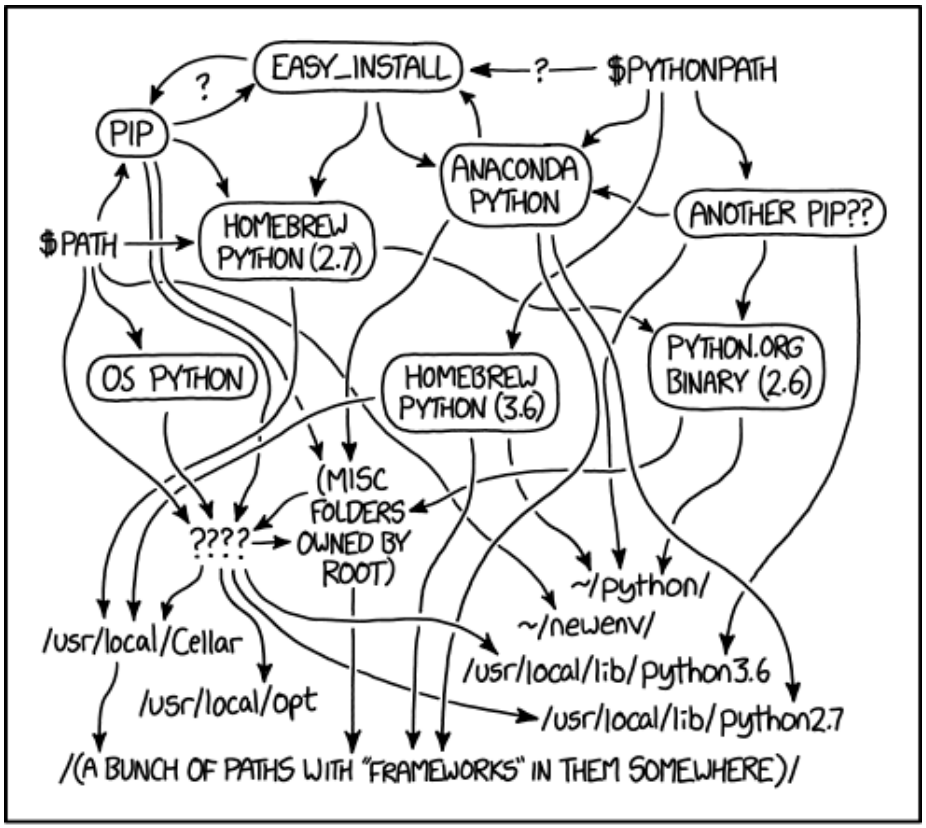


Figure 1: XKCD: Python environment

Instalaci ověříme tak, že zkusíme z příkazové řádky (konzole, terminálu, *command line* atp.) interpreta spustit.

```
python   # win
python3  # linux/macos
```

## Command line

### Windows

Nejsnáz asi v nabídce Start napsat `cmd` a potvrdit. Základní příkazy - `cd` = change directory. Měla by následovat cesta, tj.

```
C:\Users\vaclav> cd Documents\python
C:\Users\vaclav\Documents\python>
```

- `mkdir nova_slozka` - vytvoří novou složku se jménem `nova_slozka`
- `dir` vypíše obsah aktuální složky
- `del PATH\TO\FILE` vymaže soubor
- `rmdir PATH\TO\DIR` vymaže složku

### Linux/MacOS

Univerzální zkratku pro spuštění příkazové řádky neznám

- `cd` = change directory. Měla by následovat cesta, tj.

```
vaclav@pc:~ $ cd Documents/python
vaclav@pc:~/Documents/python $
```

- `mkdir nova_slozka` - vytvoří novou složku se jménem `nova_slozka`
- `ls` vypíše obsah aktuální složky
- `rm PATH/TO/FILE` vymaže soubor
- `rmdir PATH/TO/DIR` vymaže složku

## Virtuální prostředí

V praxi se může stát, že balíčky/moduly, které potřebujeme pro práci na různých projektech, jsou navzájem nekompatibilní, nebo si prostě chceme nějaký balíček vyzkoušet, aniž by zasahoval do funkčního prostředí.

K tomu slouží modul `virtualenv`, pomocí kterého můžeme vytvářet nezávislá prostředí, do nichž je možné instalovat další balíčky/moduly nezávisle. Modul `virtualenv` je nutné nainstalovat globálně, tedy v příkazové řádce:

```
pip install virtualenv   # win
pip3 install virtualenv  # linux/macos
```

Virtuální prostředí vytvoříme příkazem

```
virtualenv <my_env_name>
```

nebo

```
python -m virtualenv <my_env_name>   # win
python3 -m virtualenv <my_env_name>  # linux/mac
```

který v aktuální složce vytvoří novou složku s názvem `<my_env_name>`.

Virtuální prostředí je nutné aktivovat:

```
path\to\my_env\Scripts\activate     # win
source path/to/my_env/bin/activate # linux/macos
```

Po úspěšné aktivace se na začátku každého řádku v konzoli objeví název prostředí v závorce. Je-li prostředí aktivní, veškerá činnost související s pythonem nadále probíhá v tomto prostředí. Včetně instalace nových balíčků.

**Pozor:** platnost virtuálního prostředí je omezena na konkrétní okno příkazové řádky. Otevřu-li novou instanci příkazové řádky, žádné virtuální prostředí v ní aktivní nebude (bude chybět název v závorce).

Práci s virtuálním prostředím mohu ukončit jeho deaktivací příkazem

```
deactivate                          # linux/macos (nekdy i win)
path\to\my_env\Scripts\deactivate # win (vetsinou)
```

Pokud se ve virtuálním prostředí něco pokazí, například: - instalace nového balíčku - balíčky po aktualizaci přestanou být kompatibilní - balíček obsahuje závažné chyby působící kolaps interpreta mohu virtuální prostředí jednoduše smazat (smazaním prostředí obsahující složky) a systémová instalace zůstane nedotčena.

*Poznámka:* Je-li pro vás práce v příkazové řádce nová, zkuste vygooglit něco jako

```
how to use the command line on windows/linux/mac
```

## Jupyter a první program

Většinu věci v tomto semestru budu prezentovat ve formátu Jupyter notebooků, neboť v nich mohu snadno kombinovat funkční kód s komentářem, obrázky apod. K prohlížení Jupyter notebooků, tj. souborů ve formátu `.ipynb` je nutno prostředí Jupyter naninstalovat.

Pro zjednodušení předpokládejme, že máte ve Windows na disku C: složku python (nebo v home na linux či maxOS). Takto může vypadat postup v příkazové řádce:

### Windows

```
# presun do slozky python
C:\Users\username>cd C:\python

# nasleduji blok mozno preskocit, pokud uz virt. prostredi mame
```

```
# vytvoreni slozky pro virtualni prostredi
C:\python>mkdir venv
C:\python>cd venv
# vytvoreni virt. prostredi s nazvem env01
C:\python\venv>virtualenv env01
# navrat do predchozi slozkky
C:\python\venv>cd ..

# aktivace prostredi
C:\python>.\venv\env01\Scripts\activate
(env01) C:\python>

# instalace Jupyteru (pokud jeste nemame)
(env01) C:\python>pip install jupyter

# spusteni prostredi
(env01) C:\python>jupyter-notebook
```

**Linux** (a asi i MacOS)

```
# presun do slozky python
~ $ cd python

# nasleduji blok mozno preskocit, pokud uz virt. prostredi mame
# vytvoreni slozky pro virtualni prostredi
~/python $ mkdir venv
~/python $ cd venv
# vytvoreni virt. prostredi s nazvem env01
~/python/venv $ virtualenv env01
# navrat do predchozi slozkky
~/python/venv $ cd ..

# aktivace prostredi
~/python $ ./venv/env01/bin/activate
(env01) ~/python $

# instalace Jupyteru (pokud jeste nemame)
(env01) ~/python $ pip3 install jupyter

# spusteni prostredi
(env01) ~/python $ jupyter-notebook
```

V prostředí Jupyter-notebook tak uvidíte všechny soubory a složky, které se nachází v naší kořenové složce `python`. Stáhnete-li si tento ukázkový notebook pod názvem např. `python01.ipynb`, měli byste ho být pomocí Jupyteru schopni snadno otevřít. Můžete si v Jupyteru rovnou vyzkoušet i klasický nejjednodušší první program:

```
print("Hello world")
```

Jen pro úplnost uvádím strukturu složky python

```
python:
+---first_program.ipynb
+---python01.ipynb
+---venv
    +---env01
    +---env02
    .
    .
    .
    \---env27
```