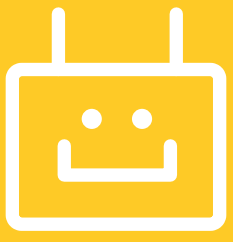
Microsoft a czechitas #NováGenerace uvádí

PROGRAMUJU

...v jazyce Python

Tutoriál – Příklady pro začátky programování v jazyce Python

Tento tutoriál je vhodný pro začátečníky od 13 let.



Co již umíš?

V předchozím tutoriálu jsme se společně seznámili s úvodem do programování v jazyce Python. Prošli jsme instalaci a odhalili, co se skrývá pod pojmy proměnná, deklarování, cyklus, podmínka (a mnoho dalších). No a jelikož se nejvíce naučíš na příkladech, přinášíme ti druhý díl – tentorkát zaměřený na maličko složitější příklady.

Co bude v tutoriálu nového?

Nejprve se vrátíme trochu na začátek a uděláme si jasno v pár dalších pojmech. Jsou jimi textový editor a příkazový řádek / příkazová řádka. Každý tuhle potvůrku nazývá trochu po svém. Podíváme se také na pár hravých věcí, naprogramujeme si třeba hru, kterou si možná ještě pamatuješ ze školních lavic. Jenže... v Pythonu nám přijde všechno trochu zábavnější.

Příkazový řádek

V minulém tutoriálu jsme se o něm již zmiňovali. Než ale začneš pořádně programovat, je potřeba vědět, co se pod kouzelným pojmem skrývá. Ať totiž chceš nebo ne, příkazový řádek bydlí i u tebe v počítači – a je lepší si z něj vychovat kamaráda než nepřítele. Jakmile si na sebe zvyknete, ušetří ti spoustu času.

Co je příkazový řádek a jak vypadá?

Jde o černé okénko, které programátorky a programátoři používají k zadávání příkazů. Na první pohled může vypadat nepřátelsky, ale to se změní.

Příkazový řádek je program, kterému se říká také konzole či terminál (anglicky command line, console, terminal) se na různých systémech otevírá různě.

Máš Windows?
Klikni na Start, napiš na klávesnici „cmd“ a vyber *Příkazový řádek* (v případě, že máš systém v češtině) nebo *Command Prompt* (jsou-li tvé Windows v angličtině)

Máš macOS?
V Aplikacích (Applications) vyber Jiné (Utilities) a zvol černou ikonu Terminal

U Linuxu hledej Konsole nebo Terminál.

Po spuštění konzole se objeví řádek, kterým počítač vybízí k zadání příkazu. Záleží na tom, jaký máš operační systém – podle něj bude končit buď znakem $ (Unix – tedy Linux či macOS) nebo > (Windows). Před tímto znakem můžou být ještě další informace.

První příkazy

Vyzkoušej si napsat první příkazy do příkazového řádku. Vyzkoušej třeba whoami (*Who am I?* znamená v angličtině *Kdo jsem?*). Napiš příkaz a potvrď jej Enterem. Jako odpověď dostaneš své přihlašovací jméno. Pokud se tvůj počítač (účet) jmenuje třeba Jarvis, uvidíš něco takového:

```
> whoami
pocitac\Jarvis
```

Znak > jsme ti schválně napsali jinou barvou. Protože ty jej nezadáváš, objeví se sám v příkazovém řádku (a před ním bude obvykle ještě text).

Adresáře

Příkazový řádek vždy pracuje v nějaké složce (jinak řečeno adresáři). Anglicky se pak hovoří o *directory* či *folder*. Je proto dobré si vždy pohlídat, v jaké složce se zrovna pohybuje. Jednoduše se ti totiž může stát, že se snažíš spustit program, který se ale nachází v jiném adresáři, než se kterým aktuálně pracuje příkazový řádek.

Název umístění adresáře, se kterám aktuálně příkazový řádek pracuje, zjistíš pomocí příkazu. Pozor, příkaz se opět liší posle tvého operačního systému.

Máš Windows? Hledáš tedy *current directory*.

```
> cd
```

Máš macOS či Linux (tedy Unix)? Hledáš *print working directory.*

```
$ pwd
```

První příkazy

Když už víš, v jakém adresáři se nacházíš, je dobré také zjistit, co se v něm všechno nachází. A přesně k tomu slouží příkaz dir, respektive ls. Vypíše ti, co vše aktuální agresář obsahuje – soubory i podadresáře.

Máš Windows? Použiješ příkaz dir *(directory)*.

```
> dir
 Directory of C:\Users\Jarvis
05/04/2018 09:22 PM <DIR>  Applications
05/04/2018 09:22 PM <DIR>  Desktop
05/04/2018 09:22 PM <DIR>  Downloads
...
```

Máš Unix? Použiješ ls *(list – tedy procházet)*

```
$ ls
Applications
Creative Cloud Files
Desktop
Documents
Downloads
...
```

Chceš se dostat někam jinam? Být totiž v adresáři je jedna věc, ale naučit se dosat i někam jinam, je něco, co určitě chceš. Uživatelé Windows mají v tomhle malou výhodu, příkaz pro změnu adresáře (tedy *change directory*) je totiž úplně stejný jako ten, který zjišťuje aktuální polohu (*current directory*). Oba mají zkratku cd. Pokud pracuješ s Unixem, tak si cd také zapamatuj, zkratka platí i pro tebe.

```
> cd Documents
> cd
C:\Users\Jarvis\Documents
```

```
$ cd Documents
$ pwd
home/Jarvis/Documents
```

Když se teď umíš pohybovat z adresáře do adresáře, zkus si v nich také vytvářet další složky (jiné adresáře). Unix i Windows mají tento příkaz zase společný, jmenuje se mkdir (make directory). Za tento příkaz napiš jméno adresáře, který chceš vytvořit.

```
> mkdir lekce
```

```
$ mkdir lekce
```

Příkazů samozřejmě existuje spousta. Bylo by bláhové se je chtít hned všechny naučit.. Mrkni proto na seznam těch, které v začátcích nejvíce využiješ.

Windows	Unix	Popis příkazu	Příklad použití
cd	cd	změna adresáře	`cd test`
cd	pwd	výpis aktuálního adresáře	`cd` `pwd`
dir	ls	výpis adresáře	`dir` `ls`
mkdir	mkdir	vytvoření adresáře	`mkdir lekce`
copy	cp	zkopírování souboru	`copy puvodni.txt kopie.txt` `cp puvodni.txt kopie.txt`
move	mv	přesun/přejmenování souboru	`move kopie.txt novemisto.txt` `mv kopie.txt novemisto.txt`
del	rm	smazání souboru	`del kopie.txt` `rm kopie.txt`
exit	exit	ukončení	`exit`

Tip! Co příkazový řádek oproti texotvému editoru neumí?
Neumí ti napovídat a nepomáhá ti hledat chyby. Takže když něco špatně zapíšeš, zjistíš chybu až po spuštění. Textový editor ti pomáhá se syntaxí daného jazyka také pomocí našeptávání. Zkus zadat část nějakého příkazu, máš-li zaplé našeptávání, pomůže ti text doplnit.

Virtuální prostředí

Co tě teď čeká? Nainstalovat Virtualvenv. Vytvoříš si virtuální prostředí pro práci v Pythonu. Instalace je na každém počítači odlišná.

> Co je virtuální prostředí?
> Virtuální prostředí zajistí, aby se tvůj počítač choval víceméně stejně, jako ostatní počítače. Znamená to, že ať již máš operační systém Windows nebo jakýkoli jiný, budeš se moci učit z těch stejných materiálů.
> Všechna rozšíření, která navíc budeš instalovat do svého virtuálního prostředí se projeví právě v tom daném prostředí. Takže i když občas uděláš nějakou chybu, vůbec neovlivní tvé další projekty.

Máš Windows?

Od minulého tutoriálu máš již nainstalovaný Python nainstalovaný. Vytvoř si teď adresář (složku), ve kterém budeš mít všechny své procvičovací úkoly k Pythonu. Může to být třeba C:\lekce. Tento adresář po vytvoření nepřesouvej jinam (jinak nebude virtuální prostředí fungovat a budeš jej muset vytvořit znovu). Už z toho důvodu jej nevytvářej na Ploše.

Až jej vytvoříš, spusť příkazový řádek a přepni se do daného adresáře pomocí příkazu cd. Pak vytvoř virtuální prostředí:

```
> py -3 -m venv venv
```

Tvůj adresář teď může vypadat ~/lekce\venv. Jsou v něm soubory s virtuálním prostředím. Můžeš se podívat dovnitř, ale nikdy tam nic neměň.

Virtuální prostředí ještě aktivuj:

```
> ~/lekce\venv\Scripts\activate
```

Na začátku příkazového řádku by se teď mělo objevit slovo venv. Znamená, že je virtuální prostředí aktivní. Tenhle příkaz budeš muset zadat vždyc, když pustíš příkazový řádek, než se pustíš do programování.

Máš macOS?

Vytvoř si teď adresář (složku), ve kterém budeš mít všechny své procvičovací úkoly k Pythonu. Může to být třeba /home/Jarvis/lekce (tedy ~/lekce). Pamatuj si, kam jej vytváříš. Nevytvářej jej na Ploše a ani jej nepřesouvej (proč zmiňujeme již u instalace pro Windows). Až adresář vytvoříš, spusť příkazový řádek a přepni se do daného adresáře pomocí příkazu cd. Pak vytvoř virtuální prostředí:

```
$ python3 -m venv venv
```

Tvůj adresář teď může vypadat ~/lekce\venv. Virtuální prostředí spustíš takto:

```
$  source ~/lekce/venv/bin/activate
```

Máš Linux?

Budeš postupovat prakticky stejně jako když máš macOS. Jen je důležité nezapomenout na jeden krok, a sice zjistit, že máš výše zmiňovaný venv. Spusť proto v příkazovém řádku příkaz:

```
$  python3 -m ensurepip -- version
```

Objeví-li se výpis začínající „pip“, nic dále neřeš. V opačném případě, objeví-li se nápis *No module named ensurepip*, je potřeba doinstalovat alternativu, Virtualenv. Jeho instalace je odlišná s každým typem Linuxu (s každou distribucí).

Tip! Instalace umí dát člověku zabrat. Mysli i tentokrát na své zdraví a jdi se protáhnout ještě před tím, než se pustíš do příkladů. Příklady v tomto tutoriálu jsou totiž trochu náročnější, než ty v minulém díle.
Měř si také čas strávený vsedě u počítače a doplňuj jej pravidelnými pohybovými aktivitami. Zaměř se také na správné sezení.

Příklady

Složitější kalkulačka

V tomto příkladě využijeme jednoduchou kalkulačku z příkladu z předchozího tutoriálu. Jednoduší variantu kalkulačky rozšíříme o možnost zadat typ operace. Použijeme tedy začátek z jednoduší varianty kalkulačky:

```
print('Vitejte v programu slozitejsi kalkulacky'

prvni = input('Zadejte prvni cislo:')
druhe = input('Zadejte druhe cislo:')
```

Nyní načteme od uživatele druh operace:

```
operator = raw_input('Jaky druh operace chcete s cisly provest?
Moznosti: +, -, *,/')
```

Teď už víme jakou operaci chce uživatel provést. Zapíšeme tedy složenou podmínku, která na základě zadaného operátoru vypočítá výsledek do proměnné **vysledek**. Pokud uživatel zadal nepodporovaný operátor, vypíšeme o tom informaci.

```
vysledek = 0;
if operatror == '+':
      vysledek = prvni + druhe
elif: operatror == '-':
      vysledek = prvni - druhe
elif: operatror == '*':
      vysledek = prvni * druhe
elif: operatror == '/':
      vysledek = prvni / druhe
else:
      print('Byl pouzit nepodporovany operator')
```

Následně vypíšeme výsledek stejně jako v případě jednoduché kalkulačky:

```
print('Vysledek je' + str(vysledek))
```

Výsledný program nalezneš na straně 9.

```
print(‘Vitejte v programu slozitejsi kalkulacky‘

prvni = input(‘Zadejte prvni cislo:‘)
druhe = input(‘Zadejte druhe cislo:‘)

operator = raw_input(‘Jaky druh operace chcete s cisly provest?
Moznosti: +, -, *, /‘)

vysledek = 0;
if operatror == ‘+‘:
      vysledek = prvni + druhe
elif: operatror == ‘-‘:
      vysledek = prvni - druhe
elif: operatror == ‘*‘:
      vysledek = prvni * druhe
elif: operatror == ‘/‘:
      vysledek = prvni / druhe
else:
      print(‘Byl pouzit nepodporovany operator‘)

print(‘Vysledek je‘ + str(vysledek))
```

Počet písmen ve slově

Napišme si program který nám spočítá písmena v zadaném slově. Pro řešení tohoto příkladu použijeme cyklus.

Přivítáme uživatele a zeptáme se ho na požadované slovo:

```
print(‘Vitejte v programu pro zjisteni delky slova‘

slovo = raw_input(‘Zadejte slovo:‘)
```

Nyní už víme požadované slovo a musíme zapsat cyklus, který projde jeho písmena a pro každé zvýší hodnotu pomocné proměnné **pocet** :

```
pocet = 0
for pismeno in slovo:
      pocet = pocet + 1
```

Nakonec pomocnou proměnnou **počet** vypíšeme jako výsledný počet písmen (opět musíme dát pozor na datové typy):

```
print(‘Slovo ‘ + slovo + ‘ má ‘ + str(pocet) + ‘ pismen‘)
```

Výsledný program:

```
print('Vitejte v programu pro zjisteni delky slova'

slovo = raw_input('Zadejte slovo:')

pocet = 0
for pismeno in slovo:
      pocet = pocet + 1

print('Slovo ' + slovo + ' má ' + str(pocet) + ' pismen'))
```

Oběšenec (hangman)

Zadání tohoto úkolu bude trochu jiné, než předchozí příklady. Jedno z možných řešení najdeš na konci celého dokumentu.

Cílem je vytvořit Python program klasické hry šibenice. Slovo, které hráč hádá, je reprezentováno pomocí řady pomlček. V případě, že hráč uhodne písmeno, které je ve slově obsaženo, program nahradí všechna písmena na na správných pozicích. Každý hráč má 10 tahů na uhodnutí slova, jinak hra končí.

Požadavky na program:

- Program v Pythonu 3.x.x
- Slovo, které se hádá, je pevně nastaveno v kódu (hardcoded)
- Počet pokusů na uhodnutí slova je také pevně nastaveno v kódu (a je vhodné, aby byl počet pokusů vyšší, než jaký je počet znaků ve slově)
- Program přivitá uživatele, zeptá se na jeho jméno a tímto jej poté oslovuje
- Program vhodně upozorňuje na průběh hry (např.: "Začínáme hádat", "Zbývá 5 pokusů", "Prohrál/Vyhrál jsi", atd..)
- Po každém kole (hádání) program vypíše výsledné slovo. Všechna neuhodnutá písmena jsou reprezentována pomlčkou, ta uhodnutá jsou nahrazena na příslušných místech
- Pokud uživatel vyčerpá pokusy, hra končí (prohrou)
- Pokud uživatel uhodne všechna písmena, hra končí (uživatel vyhrál)

Tip! Standardní print('Ahoj') vypisuje vždy na nový řádek, tudíž tak bude hra šibenice "na výšku". Pomocí zápisu print('Ahoj', end='') lze dosáhnout toho, že se budou jednotlivá písmena vypisovat vedle sebe (je potřeba pohlídat, aby se odřádkovalo vždy to poslední).
Pro vylepšení programu lze využít knihovny time . Například pro dramatickou pauzu po hádání písmena.

```python
# Zdroj: http://www.pythonforbeginners.com/code-snippets-source-code/game-hangman
# Prelozeno do cestiny a prepsano do Python 3

# naimportujeme time module
import time

# uvitejte uživatele
name = input("Jaké je je tvoje jméno? ")
print("Ahoj, " + name, ". Je čas si zahrát šibenici!")
print("")

# pockej 1 sekundu
time.sleep(1)
print("Začínáme hádat...")
time.sleep(0.5)

# tady nastavime slovo ktere budeme hadat
slovo = "secret"
# vytvorime promenou s prazdnou hodnotou pro ukladani tipu
tipy = ''
# nastavime pocet moznych pokusu
pokusy = 10

# vytvorime while cyklus
# v podmince zkontrolujeme zda jsou `pokusy` vetsi net nula
while pokusy > 0:
    # vytvorime pomocnou promenou ktera bude zacinat na nule
    neuspesne = 0

    # for cyklus pro kazde pismeno v promenné `slovo`
    for pismeno in slovo:
        # podivame se jestli je pismeno v tipech
        if pismeno in tipy:
            # vypiseme pismeno
            print(pismeno),
        else:
            # pokud neni vypiseme podtrzitko/pomlcku
            print("_"),
            # pricteme neuspesny pokus
            neuspesne += 1

    # pokud je `neuspesne` rovno nule
    if neuspesne == 0:
        # vypiseme "Vyhrali jste!"
        print("Vyhrál/a jsi!")
        # ukoncime program
        break

    print("")

    # pozadame uzivatele aby si tipnul pismeno
    tip = input("Tipni si pismeno: ")

    # pridame uzivateluv tip k ostatnim tipum
    tipy += tip

    # pokud tip neni v hadanem slove
    if tip not in slovo:
        # odecteme jeden pokus
        pokusy -= 1
        # vypiseme `Spatny tip :/`
        print("Špatný tip :/")
        # vypiseme kolik zbyva pokusu
        print("Zbývá ti ", + pokusy, ' pokusů')

    # pokud se pokusy rovnají nule
    if pokusy == 0:
        # vypiseme ze uzivatel prohral
        print("Prohrál/a jsi :(")
```

Tento materiál vznikl v rámci projektu Akademie programování, na němž spolupracují organizace Czechitas s firmou Microsoft.

Autory tutoriálu jsou Dominik Snopek a Pavla Randáková. Pavla působí v organizaci Czechitas na pozici Youth Education specialist.

Našli jste v textu nesrovnalosti? Veškeré dotazy, náměty a komentáře prosím směřujte na **paja@czechitas.cz**