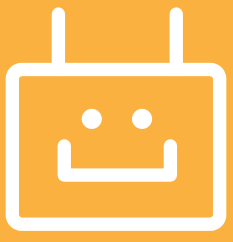
Microsoft a czechitas #NováGenerace uvádí

PROGRAMUJU

...v jazyce Python

Tutoriál pro první kroky s programovacím jazykem Python

Tento tutoriál je vhodný pro začátečníky od 13 let.



Co je Python?

Python je objektově orientovaný programovací jazyk, pomocí něhož lze vytvářet aplikace či zpracovávat data. Co znamená objektově orientovaný se dozvíš níže v textu. Jeho výhodou je mimo jiné jeho snadná syntaxe.

Budeš-li začínat s programováním právě pomocí tohoto jazyka, jednoduše si osvojíš návyky, které oceníš při práci s jakýmkoli jiným jazykem – třeba odsazování, díky nemuž bude tvůj kód mnohem přehlednější.

Co je programování?

Pomocí programování úkolujeme počítač, aby plnil úlohy, které mu zadáme. Učíme jej tedy, aby nás poslouchal a my dané úkoly nemuseli dělat sami.

Má to ale háček, možná dva. Tak jako lidé mluví na světě spoustou různých jazyků, mají své vlastní jazyky i počítače. Abychom je tedy mohli ovládat, musíme rozumět jejich řeči. Tedy stejně jako musíš v zahraničí používat cizí jazyky, je potřeba zvolit správný jazyk také pro komunikaci s technologiemi.
V čem tkví druhý háček? Zatímco lidé jsou schopni tolerovat chyby v cizí řeči a dokáží porozumět i člověku, který chybně skloňuje či časuje, počítače nic podobného nedokáží. Jakmile jim zadáš chybný příkaz, začnou si stěžovat, že mu nerozumí. Máš v něm tedy chybu, kterou je potřeba opravit a počítače úkol nesplní, dokud si ji neopravíš. Mohli bychom je tedy přirovnat k přísným paním učitelkám či pánům učitelům.

Mohli bychom také říct, že programování je vytváření postupu, které vede k řešení dané úlohy. Jde o takzvaný proces algoritmizace dané úlohy. Vytváření algorytmů je v podstatě vymyšlení postupu.

Úplně stejně, jako jsou odlišné cizí jazyky, liší se také jazyky programovací. Některé se sobě trochu podobají – něco jako čeština a slovenština. Mohli bychom říci, že mají podobnou gramatiku či skladbu věty. Jakmile se tedy naučíš pracovat s jedním jazykem, naučit se jemu podobný jazyk ti půjde snadno.

Jiné programovací jazyky jsou z úplně jiných rodin. Představ si češtinu vedle arabštiny nebo čínštiny. Tyto jazyky nemají příliš mnoho viditelných shod. Jinak se zapisují, jinak se čtou.

Každý jazyk navíc slouží k něčemu trochu jinému.

Podobnými jazyky jsou třeba:

Ruby (on Rails) a Python – oba dva jsou objektově orientované jazyky. Oba ti navíc budou připomínat, jak je důležité se učit angličtinu, protože jejich příkazy jsou anglicky.

Co znamená **objektové programování** či objektově orientované jazyky?
Když řešíš úlohu, označíš si nějaké objekty (třeba tlačítko). Každý z objektů má své *vlastnosti* a *metody*. Do kódu pak zadáváš události (*procedury*) vztahující se k těmto objektům. Uvnitř procedur pak postupuješ strukturovaně.

Strukturované programování znamená, že si složitou úlohu rozdělíš na menší úkoly, které je možné řešit samostatně. Ve strukturovaném programování se setkáš se sekvencemi, větvením a cykly.

Programovacích jazyků je spousta – a dělí se na různé typy.

Tip! Pomocí programování se člověk učí rozkládat si problémy na menší části, což je schopnost, kterou využije kdykoli. Nemusí z tebe tedy nutně vyrůst někdo, kdo se programováním živí. Zkušenosti s programováním však pro tebe budou k nezaplacení.

Začínáme programovat

Instalace Python

WINDOWS

1. Zajdi na stránky https://www.python.org/downloads (stáhnout Python 3.6.5)
2. Zaškrtni „Install launcher for All Users“ a „Add Python 3.6 to PATH“

3. Po nainstalování otevři příkazový řádek (stisk **Win** a pak napiš *cmd*)
   a) Zadej příkaz `python` nebo `python 3`
   b) Na začátku řádku se objeví `>>>` a ty můžeš zadat první příkaz v Pythonu, třeba: `2+2`, nebo `print('hello')`, příkaz potvrď ENTERem
   c) Pro ukončení Pythonu napiš příkaz `exit()`, pak pro ukončení příkazové řádky napiš znovu `exit` (nebo zavři křížkem, svět se nezboří)

Tip! ' = apostrof = alt + 39

Mac OS X

1. Nainstaluj Homebrew -- otevři terminál a zadej příkaz:
`/usr/bin/ruby -e „$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)“`
2. Po nainstalování zadej do terminálu: `brew install python3`
3. Pokračuj body 3. a) až c) u Windows

LINUX

1. Spusť v terminálu příkaz `python3 –version`
pokud se vypíše `command not found`, tak pokračuj v návodu dál
2. Zadej do terminálu: `sudo apt-get install python3`
3. Pokračuj body 3. a) až c) u Windows

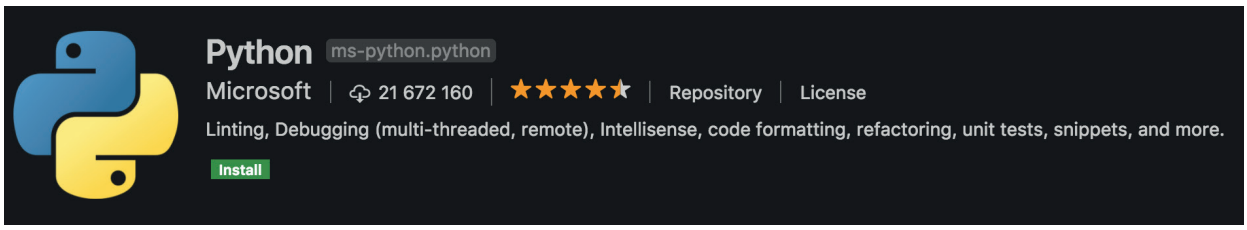
Instalace Visual Studio Code

Je potřeba někam kódy zapisovat. A jelikož programování v poznámkovém bloku není tolik zábavné, zvol spolu s námi Visual Studio Code. Je zdarma a je možné v něm programovat pomocí všemožných programovacích jazyků.

Visual Studio Code stáhni ze stránek https://code.visualstudio.com/

Stáhni si verzi podle tvého operačního systému.

Visual Studio Code po nainstalování nerozumí jazyce Python, je proto potřeba stáhnout rozšíření pro Python. Najdeš jej v **Extensions**.



Jednoduše klikni na Install. A otevři svůj textový editor, ať se s ním můžeš seznámit.

Bude ti hodně pomáhat – ihned po instalaci ti pomůže, které balíčky si máš nainstalovat.

Proměnné a hodnoty

Abychom mohli v programu například sečíst dvě čísla, budeme potřebovat tyto hodnoty nejak v programu uložit. K tomu se v programování využívají *proměnné*. Proměná se takzvaně deklaruje – musíme programu říct, jak se jmenuje, a poté se inicializuje. Inicializace znamená, že řekneme programu, jaká je její hodnota. Vše si ukážeme na jednoduchém příkladu:

```
vek = 17
```

Tímto zápisem jsme vytvořili proměnou **věk** a řekli že její hodnota je **5** . Nyní pokáždé, když budeme chtít využít proměnou **vek** stačí v programu použít její název.

Při vytváření proměnných je ale potřeba dávat pozor, jaké hodnoty do nich zapisujeme. Z tohoto důvodu v programování existuje něco čemu se říká *datové typy*.

Datové typy v jazyce Python

Python je takzvaně **dynamicky typovaný programovací jazyk**. To v praxi znamená, že pokud vytvářím proměnnou, nemusím dopředu definovat, jaký má mít datový typ.

Pokud bychom ale používali nějaký staticky typovaný jazyk, jako například Javu, museli bychom dobředu *deklarovat* datový typ proměné.

```
vek = 5
```
zápis proměné v Pythonu

```
int vek = 5
```
zápis proměné v Javě (int jako *Integer*, celé číslo)

Díky dynamickým datovým typům můžeme do proměné v Pythonu na jednom řádku uložit číslo a na druhém ho přepsat za text. Má to svoje výhody, ale jak si později ukážeme, i nevýhody.

Jaké základní datové typy existují, můžeš vidět v tabulce:

Název	Datový typ	Příklad
Integer	Celé číslo	`6`
Float	Číslo s desetinnou čárkou	`6.3`
String	Textový řetězec	`'text'`
Boolean	Logická hodnota (ano/ne)	`True` či `False`

Základní operace

Operace s čísly

S uloženými hodnotami v proměných můžeme dělat základní operace. Pokud máme číselnou hodnotu můžeme jí sčítat, odečítat, násobit:

```
>>> 1+1
2

>>> 1—1
0

>>> 8/4
2

>>> 5*2
10
```

Stejně jako v matematice musíme dát pozor na dělení nulou!

Celočíselné dělení

Další nástrahou v programování je takzvané celočíselné dělení. V podstatě to znamená, že si musíme dát pozor, jakým číslem dělíme abychom dostali správný výsledek.
Zde je praktický příklad:

```
>>> 2/3
0

>>> 2/3.0
0.6666666666666666
```

V prvním případě jsme dělili dvě celá čísla, proto jsme dostali výsledek také jedno celé číslo (0).

Bohužel jsme ale tímto zápisem přišli o informaci za desetinou čárkou. Proto jsme v druhém případě z jednoho čísla udělali číslo s desetinou čárkou, čímž jsme dostali výsledek také s desetinou čárkou (0.6666666666666666) a nepřišli tím o informaci.

Tip! Vyzkoušej, co se stane, pokud z obou celých číslic uděláš čísla s desetinnou čárkou. Nastane změna oproti předchozímu případu?

Operace s řetězci

V Pythonu můžeme používat některé operace i na textové řetezce. Pomocí operátoru + (plus) můžeme k sobě spojovat libovolné množství řetezců:

```
>>> jmeno = ‘Pavel‘
>>> vysledny_retezec = ‘Moje jmeno je , + jmeno
>>> vysledny_retezec
‘Moje jmeno je Pavel‘
```

Změna datového typu proměnné

Dostaneme se do situace, kdy bude potřeba změnit datový typ proměnné tak, abychom s ní mohli dále pracovat. Nejčasteji je potřeba převést číslo na textový řetezec nebo naopak. K tomu existují v Pythonu funkce **str()** pro převod na textový řetezec a **int()** pro převod na celé číslo.
Použití je ukázano tady:

```
>>> cislo = 12
>>> str(cislo)
‘12‘
>>> retezec = ‘12‘
>>> int(retezec)
12
```

Výstupy a vstupy

Když už umíme uložit hodnotu do proměnné a dělat s ní základní operace, naučíme se hodnoty vypisovat a načítat je od uživatele.

Výstup programu

Pro výpis informace o průběhu programu použijeme v Pythonu příkaz **print()** . Tento příkaz vypíše námi předanou hodnotu do příkazové řádky. Použití je ukázáno v následujícím příkladu:

```
>>> print('Hello world')
Hello world

>>> print (1234)
1234

>>> print(True)
True

>>> promenna = 'Toto je moje promenna'
>>> print(promenna)
Toto je moje promenna
```

Vstup programu

Abychom mohli efektivně využít potenciál programování, budeme potřebovat, aby uživatel mohl intergaovat s naším programem. K tomuto účelu existuje příkaz **input()** nebo **raw_input()** . Pomocí něho můžete načíst libovolnou hodnotu od uživatele, a tu si uložit do proměnné k dalšímu zpracování. Příkaz **raw_input()** použijeme pro načítání textových řetezců.

Po zavolání příkazu **input()** se do příkazové řádky vypíše zadaný text a program čeká na zadání hodnoty a potvrzení klávesou ENTER. Poté je zadaná hodnota uložená do příslušné proměnné a program pokračuje. Viz příklad:

```
>>> vek_uzivatele = input('Zadejte prosim vas vek:')
Zadejte prosim vas vek: 5
>>> print(vek_uzivatele)
5

>>> jmeno = raw_input('Zadejte prosim vase jmeno:')
Zadejte prosim vase jmeno: Petr
>>> print(jmeno)
Petr
```

Podmínky

Základní vlastností programů je schopnost se rozhodnout na základě vstupních podmínek. Podmínku v Pythonu zapíšeme pomocí konstrukce:

```
if <podminka>
      <toto se vykona kdyz JE podminka splnena>
else
      <toto se vykona kdyz NENI podminka splnena>
end
```

Abychom mohli zapsat podmínku, musíme znát alespoň základní operátory pro podmínky. Operátor je symbol používaný v podmínce, který např. reprezentuje porovnání dvou čísel.

Operátory

Operátory pro podmínky zpravidla dělíme na relační a logické. Relační slouží k porovnání dvou hodnot, stejně jako třeba v matematice. Logické operátory slouží ke spojení více podmínek dohromady.

Základní aritmetické operátory:

Operátor	Funkce	Použití
==	rovná se	5 == 5
>	větší než	5 > 4
<	menší než	4 < 5
>=	větší nebo rovno	5 >= 4
<=	menší nebo rovno	4 <= 5
%	modulo (zbytek po dělení)	10 % 2 == 0

Základní logické operátory:

Operátor	Funkce	Použití
and	a zároveň	<podminka1> and <podminka2>
or	nebo	<podminka1> or <podminka2>

Výsledkem podmínek je logická hodnot typu boolean (`True` nebo `False`).

Příklad použití aritmetických operátorů:

```
>>> if 5 == 5:
...   print('5 se rovna 5')
...else:
...   print('5 se nerovna 5')
...
5 se rovna 5
```

Příklad spojení 2 podmínek logickým operátorem:

```
>>> if 5 > 4 or 1 == 2 :
... print('Je pravda ze 5 je vetsi nez 4 nebo 1 se rovna 2')
... else:
... print('Neni pravda ze 5 je vetsi nez 4 nebo 1 se rovna 2')
...
Je pravda ze 5 je vetsi nez 4 nebo 1 se rovna 2
```

Výsledkem podmínky je pravda, protože při použití operátoru **or** stačí, aby jenom jedna z podmínek byla splněna. Naopak je tomu u operátoru **and,** kde je celý výraz pravda jenom tehdy, když jsou pravda i všechny jeho dílčí podmínky.

Složené podmínky

V některých případech si nevystačíme pouze s podmínkou typu „pokud ano, tak udělej tohle, a pokud ne, tak to druhé“. Někdy potřebujeme udělat složitější podmínku a proměnou testovat na více hodnot. To lze udělat přídáním klíčového slova **elif** :

```
>>> promenna = 5
>>> if promenna == 0:
...   print(,Promenna je rovna nule')
... elif promenna < 5:
...   print(,Promenna je mensi nez 5')
... else:
...   print(,Promenna je vetsi nebo rovna 5')
...
Promenna je vetsi nebo rovna 5
```

Teď si dej pauzu a jako správná programátorka / správný programátor se jdi protáhnout. Zkus také do cvičení zapojit podmínky.
Pokud sedíš u Pythonu méně než 10 minut, protažení bude trvat dvě minuty. Pokud učením trávíš mezi 10 a 20 minutami, zacvič si na 5 minut. Jinak věnuj cvičení alespoň 10 minut.
Po cvičení se mrkneme na cykly.

Cykly

Při programování se setkáme spotřebou nějakou část kódu vícekrát opakovat. Toto je možné díky cyklům. Cyklus je struktura, která nám dovolí něco opakovat podle předem stanovených podmínek. V Pythonu existují dva typy cyklů: **while** a **for**.

While cyklus

Jak už název **while** napovídá, funguje tento cyklus *zatímco (dokud)* je splněna nějaká podmínka. Obecný **while** cyklus zapíšeme takto:

```
while <podminka>:
<kod ktery se bude vykonavat dokud podminka plati>
```

Příklad programu, který vypíše čísla od 1 do 5 za použití **while** cyklu:

```
>>> cislo = 1
>>> while cislo <= 5:
...   print(cislo)
...   cislo = cislo + 1
...
1
2
3
4
5
```

For cyklus

Zatímco cyklus **while** byl vykonáván, dokud bylo splněna nějaká podmínka, **for** cyklus slouží k procházení nějkého *iterovatelného objektu*, jako je například textový řetezec. Obecně ho zapíšeme takto:

```
for <promenna>:
<kod ktery se vykona pro kazdy element objektu>
```

Příklad programu, který vypíše zadané slovo po písmenech za použití **for** cyklu:

```
>>> slovo = raw_input('Zadejte slovo:')
Zadejte slovo: test
>>> for pismeno in slovo:
...   print(pismeno)
...
t
e
s
t
```

Příklady

Je čas vyzkoušet výše sepsanou teorii v praxi.

Jednoduchá kalkulačka

Cílem programu je získat od uživatele dvě čísla, ta poté sečíst (analogicky lze použít jiný operátor) a vypsat součet do příkazové řádky.

Nejdříve uvítame uživatele v našem programu:

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

Následně načteme potřebná čísla do proměnných:

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

A nakonec obě čísla sečteme, výsledek vypíšeme do příkazové řádky. Musíme dát ale pozor na datový typ výsledku (převedeme ho na textový řetezec):

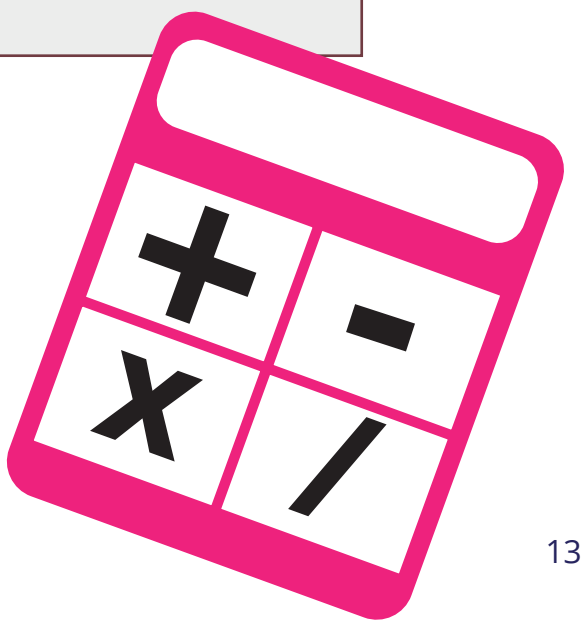
```
vysledek = prvni + druhe
print('Vysledek je' + str(vysledek))
```

Výsledný program:

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

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

vysledek = prvni + druhe
print('Vysledek je' + str(vysledek))
```



Je číslo sudé nebo liché?

Cílem programu je rozhodnout o tom, zda je zadané číslo sudé nebo liché pomocí operátoru modulo.

Nejprve opět načteme informace od uživatele:

```
print('Vitejte v programu na urceni sudeho nebo licheho cisla')
cislo = input('Zadejte cislo:')
```

Teď už stačí jenom složit podmínku, na základě které rozhodneme, zda je číslo sudé nebo liché. Operátor modulo **%** funguje tak, že vrací zbytek po dělení dvou čísel. Můžeme tedy testovat zda zbytek po dělení 2 je 0.

```
if cislo % 2 == 0:
      print('Cislo je sude')
else
      print('Cislo je liche')
```

Výsledný program:

```
print('Vitejte v programu na urceni sudeho nebo licheho cisla')
cislo = input('Zadejte cislo:')

if cislo % 2 == 0:
      print('Cislo je sude')

else
      print('Cislo je liche')
```

A to je pro začátek k Pythonu vše. Určitě také mrkni na druhý tutoriál *Příklady pro začátky programování v jazyce Python*.

Nezapomeň trénovat a zkoušet další úkoly. Zkus vymyslet složitější kalkulačky. Vypočítej třeba, kdy budou prázdniny, Vánoce nebo za jak dlouho máš narozeniny.

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**