Úvod do jazyka Python a počítačového programování

Modul 2

Vítejte ve čtvrté části! Tato část kurzu je zaměřena na proměnné - dozvíme se, co jsou to proměnné, jak je používat a jakými pravidly se řídí. Jste připraveni?

2. Sekce 4 – Proměnné

2.4.1 Proměnné – datové schránky

Zdá se poměrně samozřejmé, že Python by měl umožňovat kódování literálů nesoucích číselné a textové hodnoty.

Už víte, že s nimi můžete provádět některé aritmetické operace: sčítat, odčítat atd. To budete dělat mnohokrát.

Ale je zcela normální se ptát, jak **ukládat výsledky těchto operací**, abyste je mohli použít v dalších operacích atd.

Jak **uložit mezivýsledky a znovu je použít k vytvoření dalších výsledků**?

S tím vám pomůže jazyk Python. Nabízí k tomuto účelu speciální "boxy" (nebo "kontejnery", jak jim můžeme říkat), kterým se říká proměnné - už samotný název napovídá, že obsah těchto kontejnerů lze měnit (téměř) libovolným způsobem.

Co má každá proměnná jazyka Python?

- jméno;
- hodnotu (obsah kontejneru)

Začněme otázkami souvisejícími se jménem proměnné.

Proměnné se v programu neobjevují automaticky. Jako vývojář se musíte rozhodnout, kolik a které proměnné ve svých programech použijete.

Musíte je také pojmenovat.

2.4.2 Názvy proměnných

Pokud chcete proměnné přiřadit jméno, musíte dodržet několik přísných pravidel:

- název proměnné musí být složen z velkých nebo malých písmen, číslic a znaku _ (podtržítko).
- název proměnné musí začínat písmenem;
- znak podtržítka je písmeno;
- velká a malá písmena se považují za různá (trochu jinak než ve skutečném světě - Alice a ALICE jsou stejná křestní jména, ale v jazyce Python jsou to dvě různá jména proměnných, a tudíž dvě různé proměnné);
- název proměnné nesmí být žádné z vyhrazených slov jazyka Python (klíčová slova - o tom si brzy povíme více).

Všimněte si, že stejná omezení platí i pro názvy funkcí.

Python neklade omezení na délku názvů proměnných, ale to neznamená, že dlouhý název proměnné je vždy lepší než krátký.

Zde je několik správných, ale ne vždy vhodných názvů proměnných:

- `MyVariable`
- `i`
- `l`
- `t34`
- `Exchange_Rate`
- `counter`
- `days_to_christmas`
- `TheNameIsTooLongAndHardlyReadable`
- `_`

Tyto názvy proměnných jsou také správné:

- `Adiós_Señora`
- `sûr_la_mer`
- `Einbahnstraße`
- `переменная.`

A teď několik nesprávných názvů proměnných:

- `10t` (does not begin with a letter)
- `!important` (does not begin with a letter)
- `exchange rate` (contains a space).

Podle https://peps.python.org/pep-0008/ je doporučena následující konvence pro pojmenování proměnných a funkcí v jazyce Python:

názvy proměnných by měly být psány malými písmeny, přičemž slova by měla být oddělena podtržítkem, aby se zlepšila čitelnost (např. `var`, `my_variable`). Názvy funkcí se řídí stejnou konvencí jako názvy proměnných (např. `fun`, `my_function`). Je také možné používat smíšená velká

písmena (např. `myVariable`), ale pouze v kontextech, kde je to již převládající styl, aby byla zachována zpětná kompatibilita s přijatou konvencí.

Klíčová slova
Podívejte se na seznam slov, která hrají v každém programu v jazyce Python velmi zvláštní roli.

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal',
'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with',
'yield']
```

Říká se jim klíčová slova nebo (přesněji) vyhrazená klíčová slova. Jsou rezervovaná, protože je nesmíte používat jako názvy: ani pro své proměnné, ani pro funkce, ani pro žádné jiné pojmenované entity, které chcete vytvořit.

Význam rezervovaného slova je předdefinován a nesmí se nijak měnit.

Naštěstí vzhledem k tomu, že Python rozlišuje velká a malá písmena, můžete kterékoli z těchto slov upravit změnou velikosti kteréhokoli písmene, čímž vytvoříte nové slovo, které již není rezervované.

Například - proměnnou nemůžete pojmenovat takto:

```
import
```

Takto pojmenovanou proměnnou nesmíte mít - je to zakázáno. Místo toho však můžete udělat toto:

```
Import
```

Tato slova jsou pro vás nyní možná záhadou, ale brzy se naučíte jejich význam.

2.4.3 Jak vytvořit proměnnou

Co můžete vložit do proměnné?
Cokoliv.

Do proměnné můžete uložit jakoukoli hodnotu některého z již představených druhů a mnoho dalších, které jsme vám zatím neukázali.

Hodnota proměnné je to, co jste do ní vložili. Může se měnit tak často, jak potřebujete nebo chcete. V jednu chvíli to může být celé číslo, o chvíli později float a nakonec se z ní stane řetězec.

Nyní si povíme o dvou důležitých věcech - jak se proměnné vytvářejí a jak do nich vkládat hodnoty (nebo spíše - jak jim hodnoty dát nebo předat).

Pamatujte

Proměnná vzniká jako výsledek přiřazení hodnoty. Na rozdíl od jiných jazyků ji nemusíte nijak zvlášť deklarovat.

Pokud neexistující proměnné přiřadíte jakoukoli hodnotu, proměnná se automaticky vytvoří. Nemusíte dělat nic jiného.

Její vytvoření (jinými slovy její syntaxe) je nesmírně jednoduché: stačí použít název požadované proměnné, pak znak rovnítka (=) a hodnotu, kterou chcete do proměnné vložit.

Podívejte se na ukázku v editoru:

Skládá se ze dvou jednoduchých příkazů:

```
var = 1
print(var)
```

- První z nich vytvoří proměnnou s názvem `var` a přiřadí jí literál s celočíselnou hodnotou rovnou 1.
- Druhá vypíše hodnotu nově vytvořené proměnné do konzoly.

Jak vidíte, funkce `print()` má ještě jednu stránku - umí pracovat i s proměnnými. Víte, jaký bude výstup tohoto úryvku? Spusťte kód a zkontrolujte to.

2.4.4 Jak používat proměnnou

Můžete použít tolik deklarací proměnných, kolik potřebujete k dosažení svého cíle, například takto:

```
var = 1
account_balance = 1000.0
client_name = 'John Doe'
print(var, account_balance, client_name)
print(var)
```

Nesmíte však použít proměnnou, která neexistuje (jinými slovy proměnnou, které nebyla přiřazena hodnota).

Tento příklad způsobí chybu:

```
var = 1
print(Var)
```

Víte proč? Pokusili jsme se použít proměnnou s názvem Var, která nemá žádnou hodnotu (pozn. `var` a `Var` jsou různé entity a nemají spolu nic společného, pokud jde o Python).

Pamatujte

Pro výstup řetězců a proměnných můžete použít funkci `print()` a kombinovat text a proměnné pomocí operátoru `+`. Například:

```
var = "3.8.5"
print("Verze Pythonu: " + var)
```

Uhodnete, jaký bude výstup výše uvedeného úryvku?

2.4.5 Jak přiřadit novou hodnotu již existující proměnné

Jak přiřadíte novou hodnotu již existující proměnné? Stejným způsobem. Stačí použít znaménko rovnosti.

```
var = 1
print(var)
var = var + 1
print(var)
```

Kód odešle na konzolu dva řádky:

```
1
2
```

První řádek tohoto úryvku vytvoří novou proměnnou s názvem var a přiřadí jí hodnotu 1.

Příkaz zní: Přiřaďte proměnné s názvem var hodnotu 1.

Můžeme to říci i kratší: přiřaď proměnné var hodnotu 1.

Někteří dávají přednost čtení takového příkazu jako: var se stane 1.

Třetí řádek přiřadí téže proměnné novou hodnotu převzatou ze samotné proměnné, sečtenou s 1. Při pohledu na takový zápis by matematik asi protestoval - žádná hodnota se nesmí rovnat sama sobě plus jedna. Jedná se o rozpor. Python však znak = nepovažuje za rovná se, ale za přiřazení hodnoty.

Jak tedy takový záznam v programu přečíst?

Vezměte aktuální hodnotu proměnné var, přičtěte k ní jedničku a výsledek uložte do proměnné var.

V podstatě se hodnota proměnné var zvětšila (byla **inkrementována**) o jedničku, což nemá nic společného s porovnáváním proměnné s nějakou hodnotou.

Víte, jaký bude výstup následujícího úryvku?

```
var = 100
var = 200 + 300
print(var)
```

Zkontrolujte.

2.4.6 Řešení jednoduchých matematických úloh

Nyní byste měli být schopni sestavit krátký program řešící jednoduché matematické úlohy, například Pythagorovu větu:

```
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5
print("c =", c)
```

Poznámka: pro vyhodnocení odmocniny musíme použít operátor ** jako:

$$\sqrt{(x)} = x^{(1/2)}$$

a

$$c = \sqrt{a^2 + b^2}$$

Dokážete odhadnout výstup kódu v konzoli?

```
c = 5.0
```

2.4.7 LAB Proměnné

Scénář
Zde je krátký příběh:

Kdysi dávno v zemi jablek měl Jan tři jablka, Marie pět jablek a Adam šest jablek. Všichni byli velmi šťastní a žili dlouho. Konec příběhu.

Vaším úkolem je:

- vytvořit proměnné: `John`, `Mary` a `Adam`;
- přiřadit proměnným hodnoty. Hodnoty se musí rovnat počtu plodů, které mají `John`, `Mary` a `Adam`;
- po uložení čísel do proměnných vypište proměnné na jeden řádek a každou z nich oddělte čárkou;
- nyní vytvořte novou proměnnou s názvem `total_apples`, která se bude rovnat součtu tří předchozích proměnných;
- hodnotu uloženou v proměnné `total_apples` vypište do konzoly;
- experimentujte s kódem: vytvářejte nové proměnné, přiřazujte jim různé hodnoty a provádějte s nimi různé aritmetické operace (např. `+`, `-`, `*`, `/`, `//` atd.). Zkuste vypsat na jeden řádek řetězec a celé číslo dohromady, např. "Celkový počet jablek:" a `total_apples`.

2.4.8 Složené operátory (zkrácené operátory)

Je čas na další sadu operátorů, které vývojářům usnadňují život. Velmi často chceme použít jednu a tutéž proměnnou jak na pravé, tak na levé straně operátoru `=`.

Pokud například potřebujeme vypočítat řadu po sobě jdoucích hodnot mocnin 2, můžeme použít takovýto kousek kódu:

```
x = x * 2
```

Takový výraz můžete použít, pokud nemůžete usnout a snažíte se s tím vypořádat pomocí starých dobrých metod:

```
sheep = sheep + 1
```

Python nabízí zkrácený způsob zápisu takovýchto operací, které lze nakódovat následovně:

```
x *= 2
sheep += 1
```

Pokusme se uvést obecný popis těchto operací. Pokud je operátor *op* dvouargumentový (to je velmi důležitá podmínka) a operátor je použit v následujícím kontextu...:

```
proměnná = proměnná op výraz
```

...pak jej lze zjednodušit a zobrazit takto:

```
proměnná op= výraz
```

Podívejte se na příklady níže. Ujistěte se, že jim všem rozumíte.

Výraz	Zkrácený operátor
`i = i + 2 * j`	`i += 2 * j`
`var = var / 2`	`var /= 2`
`rem = rem % 10`	`rem %= 10`
`j = j - (i + var + rem)`	`j -= (i + var + rem)`
`x = x ** 2`	`x **= 2`

2.4.9 LAB Proměnné - jednoduchý převodník

Scénář

Míle a kilometry jsou jednotky délky nebo vzdálenosti.

Mějte na paměti, že 1 míle se rovná přibližně 1,61 kilometru, a doplňte program v editoru tak, aby převedl:

- míle na kilometry;
- kilometry na míle.

```
kilometers = 12.25
miles = 7.38

miles_to_kilometers = ###
kilometers_to_miles = ###

print(miles, "miles is", round(miles_to_kilometers, 2), "kilometers")
print(kilometers, "kilometers is", round(kilometers_to_miles, 2),
"miles")
```

Ve stávajícím kódu nic neměňte. Kód napište na místa označená `###`. Otestujte svůj program s údaji, které jsme uvedli ve zdrojovém kódu.

Zvláštní pozornost věnujte tomu, co se děje uvnitř funkce `print()`. Analyzujte, jak funkci poskytujeme více argumentů a jak vypisujeme očekávaná data.

Všimněte si, že některé z argumentů uvnitř funkce `print()` jsou řetězce (např. "mil je", zatímco jiné jsou proměnné (např. `mil`).

Tip

Děje se zde ještě jedna zajímavá věc. Vidíte uvnitř funkce `print()` další funkci? Je to funkce `round()`. Jejím úkolem je zaokrouhlit vypsaný výsledek na počet desetinných míst uvedených v závorce a vrátit float (uvnitř funkce `round()` najdete název proměnné, čárku a počet desetinných míst, o který nám jde). O funkcích si budeme povídat velmi brzy, takže se nemusíte bát, že by vám vše ještě nebylo úplně jasné. Chceme jen podnítit vaši zvědavost.

Po dokončení laboratoře otevřete experimentujte dál. Zkuste napsat různé převodníky, např. převodník z USD na EUR, převodník teploty atd. - Popusťte uzdu své fantazii! Zkuste vypsat výsledky kombinací řetězců a proměnných. Zkuste použít funkci `round()` a experimentovat s ní, abyste výsledky zaokrouhlili na jedno, dvě nebo tři desetinná místa. Vyzkoušejte, co se stane, pokud neuvedete žádný počet číslic. Nezapomeňte své programy testovat.

Experimentujte, vyvozujte závěry a učte se. Buďte zvědaví.

Očekávaný výstup:

```
7,38 míle je 11,88 kilometru
12,25 kilometrů je 7,61 mil
```

2.4.10 LAB Operátory a výrazy

Scénář

Podívejte se na následující kód:

```
x =  # Hardcode your test data here.
x = float(x)
# Write your code here.
print("y =", y)
```

Kód přiřadí natvrdo hodnotu float do proměnné `x` a vypíše hodnotu proměnné `y`. Vaším úkolem je doplnit kód tak, aby vyhodnotil následující výraz:

$$3x^3 - 2x^2 + 3x - 1$$

Výsledek by měl být přiřazen proměnné `y`.

Nezapomeňte, že klasický algebraický zápis rád vynechává operátor násobení - je třeba jej explicitně použít. Všimněte si, jak měníme datový typ, abychom se ujistili, že `x` je typu `float`.

Udržujte kód čistý a čitelný a vyzkoušejte jej pomocí námi poskytnutých dat, přičemž je pokaždé přiřaďte proměnné `x` (natvrdo). Nenechte se odradit případnými počátečními neúspěchy. Buďte vytrvalí a zvídaví.

Sample input

```
x = 0
x = 1
x = -1
```

Sample output

Output

```
y = -1.0
y = 3.0
y = -9.0
```

2.4.11 SHRNUTÍ SEKCE

1. Proměnná je pojmenované místo vyhrazené pro ukládání hodnot v paměti. Proměnná je vytvořena nebo inicializována automaticky při prvním přiřazení hodnoty. (2.1.4.1)
2. Každá proměnná musí mít jedinečný název - identifikátor. Legální název identifikátoru musí být neprázdná posloupnost znaků, musí začínat podtržítkem(_) nebo písmenem a nesmí to být *klíčové slovo* jazyka Python. Za prvním znakem může následovat podtržítko, písmeno a číslice. U identifikátorů v jazyce Python se rozlišují malá a velká písmena.
3. Python je dynamicky typovaný jazyk, což znamená, že v něm nemusíte deklarovat proměnné. (2.1.4.3) K přiřazení hodnot proměnným můžete použít jednoduchý operátor přiřazení ve tvaru znaku rovnosti (=), tj. `var = 1`.
4. K úpravě hodnot přiřazených proměnným můžete použít také složené operátory přiřazení (zkratkové operátory), například: `var += 1` nebo `var /= 5 * 2`.
5. Již existujícím proměnným můžete přiřadit nové hodnoty pomocí operátoru přiřazení nebo některého ze složených operátorů, např:

```
var = 2
print(var)
var = 3
print(var)
var += 1
print(var)
```

Pomocí operátoru + můžete kombinovat text a proměnné a pomocí funkce `print()` můžete například vypisovat řetězce a proměnné:

```
var = "007"
print("Agent " + var)
```

2.4.12 ODDÍLOVÝ KVÍZ

Otázka 1: Jaký je výstup následujícího kódu?

```
var = 2
var = 3
print(var)
```

```
a) 1
b) 2
c) 3
```

Otázka 2: Které z následujících názvů proměnných jsou v jazyce Python nelegální? (Vyberte tři odpovědi)

```
a) my_var
b) m
c) 101
d) averylongVariablename
e) m101
f) m 101
```

```
g) Del
h) del
```

Otázka 3: Jaký je výstup následujícího úryvku kódu?

```
a = '1'
b = "1"
print(a + b)
```

Otázka 4: Jaký je výstup následujícího úryvku kódu?

```
a = 6
b = 3
a /= 2 * b
print(a)
```

Odkazy:

Cisco Programming Essentials in Python

Root.cz

ITNetwork.cz

Internet