Úvod do jazyka Python a počítačového programování

Modul 2

Datové typy, proměnné, základní vstupně-výstupní operace, základní operátory

V tomto modulu se naučíte:

- jak psát a spouštět jednoduché programy v jazyce Python;
- co jsou to literály, operátory a výrazy jazyka Python;
- co jsou proměnné a jakými pravidly se řídí;
- jak provádět základní vstupní a výstupní operace.

2.1.1.1 Váš první program

Hello, World!

Je čas začít psát skutečný, funkční kód v jazyce Python. Prozatím to bude velmi jednoduché. Protože si ukážeme některé základní pojmy a termíny, nebudou tyto úryvky kódu seriózní ani složité. Spusťte následující kód v okně editoru:

```
print("Hello, World!")
```

Pokud zde vše proběhne v pořádku, uvidíte v okně konzoly řádek textu. Nyní vám ukážeme a vysvětlíme, co vlastně vidíte a proč to tak vypadá. Jak vidíte, první program se skládá z následujících částí:

slovo print;

úvodní závorky;

uvozovky;

řádek textu: Hello, World!;

další uvozovky;

uzavírací závorka.

Každý z výše uvedených znaků hraje v kódu velmi důležitou roli.

Funkce print()

Podívejte se na řádek kódu níže:

```
print("Hello, World!")
```

Slovo print, které zde vidíte, je název funkce. To však neznamená, že kdekoli se toto slovo objeví, je to vždy název funkce. Význam slova vychází z kontextu, ve kterém bylo slovo použito.

S pojmem funkce jste se pravděpodobně setkali již mnohokrát, a to v hodinách matematiky. Pravděpodobně také dokážete vyjmenovat několik názvů matematických funkcí, například sinus nebo logaritmus.

Funkce jazyka Python jsou však flexibilnější a mohou obsahovat více obsahu než jejich matematičtí sourozenci.

Funkce (v tomto kontextu) je samostatná část počítačového kódu schopná:

- způsobit nějaký efekt (např. poslat text na terminál, vytvořit soubor, nakreslit obrázek, přehrát zvuk atd.); to je ve světě matematiky něco zcela neslýchaného;
- vyhodnotit nějakou hodnotu (např. odmocninu z nějaké hodnoty nebo délku daného textu) a vrátit ji jako výsledek funkce; to je to, co dělá z funkcí jazyka Python příbuzné matematických pojmů.

Mnoho funkcí jazyka Python navíc dokáže výše uvedené dvě věci provádět společně.

Odkud se berou funkce?

- Mohou pocházet ze samotného Pythonu; jednou z takových funkcí je funkce print; taková funkce je přidanou hodnotou získanou spolu s Pythonem a jeho prostředím (je vestavěná); pokud ji chcete využít, nemusíte dělat nic zvláštního (např. se někoho na něco ptát);
- mohou pocházet z jednoho nebo více doplňků Pythonu nazvaných moduly; některé moduly jsou dodávány s Pythonem, jiné mohou vyžadovat samostatnou instalaci - ať už je to jakkoli, všechny je třeba explicitně propojit s vaším kódem (brzy si ukážeme, jak to udělat);
- můžete si je napsat sami a umístit do svého programu tolik funkcí, kolik chcete a potřebujete, aby byl jednodušší, přehlednější a elegantnější.

Název funkce by měl být **příznačný** (název funkce print je samozřejmý).

Samozřejmě, pokud se chystáte využít nějakou již existující funkci, nemáte na její název žádný vliv, ale když začnete psát vlastní funkce, měli byste výběr názvů pečlivě zvážit.

Jak jsme již uvedli, funkce může mít:

- účinek;
- výsledek.

Existuje také třetí, velmi důležitá složka funkce - **argument**(y).

Matematické funkce obvykle přijímají jeden argument, např. funkce sin(x) přijímá x, což je míra úhlu.

Naproti tomu funkce jazyka Python jsou univerzálnější. V závislosti na individuálních potřebách mohou přijímat libovolný počet argumentů - tolik, kolik je jich potřeba k plnění jejich úkolů. Poznámka: libovolný počet zahrnuje i nulu - některé funkce Pythonu nepotřebují žádný argument.

```
print("Ahoj, světe!")
```

Navzdory počtu potřebných/poskytovaných argumentů funkce Pythonu důrazně vyžadují přítomnost **dvojice závorek** - otevírací, resp. uzavírací.

Pokud chcete funkci dodat jeden nebo více argumentů, **umístíte je do závorek**. Pokud se chystáte použít funkci, která nepřijímá žádný argument, musíte závorky stále mít.

Poznámka:

abyste odlišili obyčejná slova od názvů funkcí, umístěte za jejich názvy **dvojici prázdných závorek**, i když příslušná funkce chce jeden nebo více argumentů. Jedná se o standardní konvenci.

Funkce, o které zde mluvíme, je `print()`.

Má funkce `print()` v našem příkladu nějaké argumenty?

Samozřejmě že má, ale jaké jsou to argumenty?

Jediným argumentem funkce `print()` v tomto příkladu je řetězec:

```
print("Hello, World!")
```

Jak vidíte, řetězec je ohraničen uvozovkami - uvozovky vlastně tvoří řetězec - vyřezávají část kódu a přiřazují mu jiný význam.

Můžete si představit, že uvozovky říkají něco jako: Text mezi námi není kód. Není určen k provedení a měli byste ho brát tak, jak je.

Téměř cokoli, co vložíte do uvozovek, bude bráno doslova, nikoli jako kód, ale jako data. Zkuste si s tímto konkrétním řetězcem pohrát - upravte jej, zadejte nějaký nový obsah, odstraňte část stávajícího obsahu.

V kódu Pythonu existuje více způsobů, jak zadat řetězec, ale pro tuto chvíli stačí tento.

Dosud jste se seznámili se dvěma důležitými částmi kódu: funkcí a řetězcem. Mluvili jsme o nich z hlediska syntaxe, ale nyní je čas probrat je z hlediska sémantiky.

Název funkce (v tomto případě `print`) spolu se závorkami a argumentem (argumenty) tvoří volání funkce.

```
print("Hello, World!")
```

Brzy si to probereme podrobněji, ale teď bychom si to měli trochu osvětlit.

Co se stane, když Python narazí na volání, jako je toto níže?

```
function_name(argument)
```

Podívejme se na to:

- Python nejprve zkontroluje, zda je zadaný název legální (prohledá svá interní data, aby našel existující funkci s tímto názvem; pokud toto hledání selže, Python přeruší kód);
- za druhé Python zkontroluje, zda požadavky funkce na počet argumentů umožňují funkci tímto způsobem vyvolat (např. pokud konkrétní funkce vyžaduje přesně dva argumenty, bude

jakékoli vyvolání dodávající pouze jeden argument považováno za chybné a přeruší provádění kódu);
- za třetí, Python na chvíli opustí váš kód a skočí do funkce, kterou chcete vyvolat; samozřejmě si vezme i váš argument (argumenty) a předá ho (je) funkci;
- za čtvrté, funkce provede svůj kód, vyvolá požadovaný efekt (pokud existuje), vyhodnotí požadovaný výsledek (pokud existuje) a dokončí svůj úkol;
- nakonec se Python vrátí do vašeho kódu (na místo těsně za voláním) a pokračuje v jeho provádění.

LAB 1

Odhadovaný čas

5-10 minut

Úroveň obtížnosti

Velmi snadné

Cíle

- seznámení se s funkcí print() a jejími možnostmi formátování;
- experimentování s kódem Pythonu.

Scénář

Příkaz print(), který je jednou z nejjednodušších direktiv v jazyce Python, jednoduše vytiskne řádek na obrazovku.

V prvním labu:

- použijte funkci print() k vypsání řádku Hello, Python! na obrazovku. Kolem řetězce použijte dvojité uvozovky;
- Po provedení tohoto úkonu použijte funkci print() znovu, ale tentokrát vypište své křestní jméno;
- odstraňte dvojité uvozovky a spusťte kód. Sledujte reakci Pythonu. Jaký druh chyby vyhodí?
- Potom odstraňte závorky, vraťte zpět dvojité uvozovky a znovu spusťte kód. Jaký druh chyby se vyhodí tentokrát?
- Experimentujte, jak jen můžete. Změňte dvojité uvozovky na jednoduché, použijte více funkcí print() na stejném řádku a pak na různých řádcích. Zjistěte, co se stane.

Je třeba co nejdříve zodpovědět tři důležité otázky:

1. Jaký efekt způsobuje funkce `print()`?

Efekt je velmi užitečný a velmi významný. Funkce:

přebírá své argumenty (může přijmout více než jeden argument a může také přijmout méně než jeden argument).

v případě potřeby je převede do lidsky čitelné podoby (jak možná tušíte, řetězce tuto činnost nevyžadují, protože řetězec je již čitelný)

a výsledná data odešle na výstupní zařízení (obvykle konzoli); jinými slovy, cokoli vložíte do funkce `print()`, se objeví na obrazovce.

Není tedy divu, že od této chvíle budete funkci `print()` využívat velmi intenzivně, abyste viděli výsledky svých operací a vyhodnocení.

2. Jaké argumenty očekává funkce `print()`?

Jakékoli. Brzy si ukážeme, že print() umí pracovat prakticky se všemi typy dat, které Python nabízí. Řetězce, čísla, znaky, logické hodnoty, objekty - kterýkoli z nich lze úspěšně předat funkci `print()`.

3. Jakou hodnotu vrací funkce `print()`?

Žádnou. Její účinek je dostatečný.

Funkce print() - návod

Již jste viděli počítačový program, který obsahuje jedno volání funkce. Volání funkce je jedním z mnoha možných druhů instrukcí jazyka Python.

Každý složitý program samozřejmě obvykle obsahuje mnohem více instrukcí než jednu. Otázka zní: jak do kódu jazyka Python spojit více než jednu instrukci?

Syntaxe jazyka Python je v této oblasti poměrně specifická. Na rozdíl od většiny programovacích jazyků Python vyžaduje, aby v jednom řádku nemohla být více než jedna instrukce.

Řádek může být prázdný (tj. nesmí obsahovat vůbec žádnou instrukci), ale nesmí obsahovat dvě, tři nebo více instrukcí. To je přísně zakázáno.

Poznámka:

Python dělá z tohoto pravidla jednu výjimku - umožňuje, aby se jedna instrukce rozprostřela na více než jeden řádek (což může být užitečné, pokud váš kód obsahuje složité konstrukce).

Rozšiřme trochu kód, který si můžete prohlédnout v editoru. Spusťte jej a všimněte si, co vidíte v konzoli.

```
print("The itsy bitsy spider climbed up the waterspout.")
print("Down came the rain and washed the spider out.")
```

Vaše konzola Pythonu by nyní měla vypadat takto:

```
The itsy bitsy spider climbed up the waterpout.
Down came the rain and washed the spider out.
```

To je dobrá příležitost k několika postřehům:

- **Program volá funkci** `print()` **dvakrát** a v konzoli můžete vidět dva samostatné řádky - to znamená, že funkce `print()` začíná svůj výstup pokaždé od nového řádku; toto chování můžete změnit, ale také ho můžete využít ve svůj prospěch;
- každé volání funkce `print()` obsahuje jako argument jiný řetězec a obsah konzoly to odráží - to znamená, že **instrukce v kódu se provádějí ve stejném pořadí, v jakém byly umístěny ve zdrojovém souboru**; žádná další instrukce se neprovede, dokud není dokončena ta předchozí (z tohoto pravidla existují výjimky, ale ty můžete prozatím ignorovat)

Příklad jsme trochu pozměňme - přidejme jsme jedno prázdné volání funkce `print()`. Nazýváme ji prázdnou funkcí, protože jsme funkci nedodali žádné argumenty:

```
print("The itsy bitsy spider climbed up the waterspout.")

print()

print("Down came the rain and washed the spider out.")
```

Můžete si ji prohlédnout v okně editoru. Spusťte kód.

Co se stane?

Pokud vše proběhne správně, měli byste vidět něco takového:

```
The itsy bitsy spider climbed up the waterspout.

Down came the rain and washed the spider out.
```

Jak vidíte, prázdné volání funkce `print()` není tak prázdné, jak jste možná očekávali - skutečně vypíše prázdný řádek, nebo (tato interpretace je také správná) jeho výstupem je pouze nový řádek.

To však není jediný způsob, jak ve výstupní konzole vytvořit nový řádek. Nyní si ukážeme jiný způsob.

Funkce print() - znaky escape a nový řádek

Kód jsme opět upravili. Podívejte se na něj pozorně:

```
print("The itsy bitsy spider\nclimbed up the waterspout.")

print()

print("Down came the rain\nand washed the spider out.")
```

Jsou zde dvě velmi jemné změny - vložili jsme zvláštní dvojici znaků dovnitř rámečku. Vypadají následovně: \n.

Zajímavé je, že zatímco **vy vidíte dva znaky, Python vidí jeden**.

Zpětné lomítko (\) má při použití uvnitř řetězců velmi zvláštní význam - říká se mu **escape znak**.

Slovo *escape* je třeba chápat specificky - znamená, že řada znaků v řetězci na okamžik (velmi krátký okamžik) unikne, aby zavedla speciální začlenění.

Jinými slovy, zpětné lomítko samo o sobě nic neznamená, ale je pouze jakýmsi oznámením, že další znak za zpětným lomítkem má také jiný význam.

Písmeno `n` umístěné za zpětným lomítkem pochází ze slova *newline*.

Zpětné lomítko i písmeno n tvoří speciální symbol s názvem znak nového řádku, který konzoli nabádá k zahájení nového výstupního řádku.

Spusťte kód. Vaše konzola by nyní měla vypadat takto:

```
The itsy bitsy spider
climbed up the waterspout.

Down came the rain
and washed the spider out.
```

Jak vidíte, v říkance se objevují dva nové řádky, a to v místech, kde bylo použito `\n`.

Tato konvence má dva důležité důsledky:

1. Pokud chcete do řetězce vložit jen jedno zpětné lomítko, nezapomeňte na jeho escapovací charakter - musíte ho zdvojit, např. takové volání způsobí chybu:

```
print("\")
```

zatímco toto chybu nezpůsobí:

```
print("\\")
```

2. Ne všechny escape páry (zpětné lomítko ve spojení s jiným znakem) něco znamenají.

Experimentujte se svým kódem v editoru, spusťte ho a uvidíte, co se stane.

Funkce print() - použití více argumentů

Dosud jsme testovali chování funkce print() bez argumentů a s jedním argumentem. Stojí za to také vyzkoušet nakrmit funkci print() více než jedním argumentem.

Podívejte se na okno editoru. Právě to budeme nyní testovat:

```
print("The itsy bitsy spider" , "climbed up" , "the waterspout.")
```

Existuje jedno volání funkce `print()`, které však obsahuje tři argumenty. Všechny jsou řetězce. Argumenty jsou odděleny čárkami. Obklopili jsme je mezerami, aby byly lépe viditelné, ale ve skutečnosti to není nutné a my to už dělat nebudeme.

V tomto případě hrají čárky oddělující argumenty úplně jinou roli než čárka uvnitř řetězce. Ta první je součástí syntaxe jazyka Python, ta druhá je určena k zobrazení v konzoli.

Pokud se na kód podíváte znovu, uvidíte, že uvnitř řetězců nejsou žádné mezery.

Spusťte kód a podívejte se, co se stane. V konzoli by se nyní měl zobrazit následující text:

```
The itsy bitsy spider climbed up the waterspout.
```

Mezery odstraněné z řetězců se opět objevily. Můžete vysvětlit proč?

Z tohoto příkladu vyplývají dva závěry:

- funkce `print()` vyvolaná s více než jedním argumentem je všechny vypíše na jeden řádek;
- funkce `print()` z vlastní iniciativy vkládá mezi vypisované argumenty mezeru.

Funkce print() - poziční způsob předávání argumentů

Nyní, když už víte něco o zvyklostech funkce `print()`, si ukážeme, jak je změnit. Měli byste být schopni předvídat výstup, aniž byste kód spouštěli v editoru.

Způsob, jakým předáváme argumenty do funkce `print()`, je v Pythonu nejběžnější a nazývá se **poziční způsob** (tento název pochází z toho, že význam argumentu je dán jeho pozicí, např. druhý argument bude vypsán za prvním, nikoli naopak).

Spusťte kód a zkontrolujte, zda výstup odpovídá vašim předpokladům.

```
print("My name is", "Python.")
print("Monty Python.")
```

Funkce print() - argumenty klíčových slov

Python nabízí další mechanismus pro předávání argumentů, který může být užitečný, pokud chcete funkci `print()` přesvědčit, aby trochu změnila své chování.

Nebudeme jej nyní vysvětlovat do hloubky. Plánujeme to udělat, až budeme hovořit o funkcích. Prozatím vám chceme pouze ukázat, jak to funguje. Nebojte se ji použít ve svých vlastních programech.

Mechanismus se nazývá **argumenty klíčových slov**. Název vychází ze skutečnosti, že význam těchto argumentů se nepřebírá z jejich umístění (pozice), ale ze speciálního slova (klíčového slova), které se používá k jejich identifikaci.

Funkce `print()` má dva argumenty klíčových slov, které můžete použít pro své účely. První z nich se jmenuje end.

V okně editoru vidíte velmi jednoduchý příklad použití argumentu klíčového slova.

```
print("My name is", "Python.", end=" ")
print("Monty Python.")
```

Pro jeho použití je nutné znát některá pravidla:

- argument klíčového slova se skládá ze tří prvků: **klíčového slova** identifikujícího argument (zde `end`), **znaku rovnosti** (`=`) a **hodnoty** přiřazené tomuto argumentu;
- všechny argumenty s klíčovým slovem musí být uvedeny **za posledním pozičním argumentem** (to je velmi důležité).

V našem příkladu jsme využili argumentu klíčového slova `end` a nastavili jej na řetězec obsahující jednu mezeru.

Spusťte kód a podívejte se, jak funguje.

Konzola by nyní měla zobrazovat následující text:

```
My name is Python. Monty Python.
```

Jak vidíte, argument klíčového slova `end` určuje znaky, které funkce `print()` odešle na výstup, jakmile dosáhne konce svých pozičních argumentů.

Výchozí chování odráží situaci, kdy je argument klíčového slova end **implicitně** použit následujícím způsobem: `end="\n"`.

A nyní je čas vyzkoušet něco složitějšího.

Když se pozorně podíváte, zjistíte, že jsme použili argument `end`, ale řetězec, který je mu přiřazen, je prázdný (neobsahuje vůbec žádné znaky).

```
print("My name is ", end="")
print("Monty Python.")
```

Co se stane nyní? Spusťte program v editoru a zjistěte to.

Protože argument end byl nastaven na nic, funkce print() po vyčerpání svých pozičních argumentů také nic nevypíše.

Konzola by nyní měla zobrazovat následující text:

```
My name is Monty Python.
```

Poznámka: na výstup nebyly odeslány žádné nové řádky.

Řetězec přiřazený argumentu klíčového slova `end` může mít libovolnou délku. Pokud chcete, experimentujte s ním.

Již dříve jsme si řekli, že funkce print() odděluje své výstupní argumenty mezerami. I toto chování lze změnit.

Klíčové slovo argumentu, které to umí, se jmenuje `sep` (jako oddělovač).

Podívejte se na kód v editoru a spusťte jej.

```
print("My", "name", "is", "Monty", "Python.", sep="-")
```

Argument sep přináší následující výsledky:

```
My-name-is-Monty-Python.
```

Funkce `print()` nyní používá k oddělení vypisovaných argumentů místo mezery pomlčku.

Poznámka: hodnotou argumentu `sep` může být i prázdný řetězec. Vyzkoušejte si to sami.

Oba argumenty klíčových slov mohou být smíšeny v jednom volání, stejně jako zde v okně editoru.

Příklad nedává příliš smysl, ale viditelně prezentuje interakce mezi `end` a `sep`.

```
print("My", "name", "is", sep="_", end="*")
print("Monty", "Python.", sep="*", end="*\n")
```

Dokážete předpovědět výstup?

```
My_name_is*Monty*Python.*
```

Spusťte kód a podívejte se, zda odpovídá vašim předpovědím.

Nyní, když jste pochopili funkci `print()`, jste připraveni uvažovat o tom, jak ukládat a zpracovávat data v jazyce Python.

Bez funkce `print()` byste nemohli vidět žádné výsledky.

LAB

Odhadovaný čas

5-10 minut

Úroveň obtížnosti

Velmi snadné

Cíle

seznámení se s funkcí print() a jejími možnostmi formátování;

experimentování s kódem Pythonu.

Scénář

Upravte první řádek kódu v editoru pomocí klíčových slov sep a end tak, aby odpovídal očekávanému výstupu. Použijte v editoru dvě funkce `print()`.

```
print("Programming","Essentials","in")
print("Python")
```

Ve druhém volání funkce `print()` nic neměňte.

Očekávaný výstup

```
Programming***Essentials***in...Python
```

LAB

Odhadovaný čas

5-15 minut

Úroveň obtížnosti

Snadná

Cíle

experimentování s existujícím kódem Pythonu;

odhalení a oprava základních syntaktických chyb;

seznámení se s funkcí `print()` a jejími možnostmi formátování.

Scénář

```
print("    *")
print("   * *")
print("  *   *")
print(" *     *")
print("***   ***")
print("  *   *")
print("  *   *")
print("  *****")
```

Důrazně vám doporučujeme, abyste si pohráli s kódem, který jsme pro vás napsali, a provedli v něm některé (možná i destruktivní) změny. Nebojte se upravit jakoukoli část kódu, ale je tu jedna podmínka - učte se z chyb a vyvozujte vlastní závěry.

Pokuste se:

- minimalizovat počet volání funkce `print()` tím, že do řetězců vložíte sekvenci `\n`.
- zvětšit šipku na dvojnásobek (ale zachovat její proporce)
- šipku zduplikovat a umístit obě šipky vedle sebe; poznámka: řetězec lze násobit pomocí následujícího triku: `"string" * 2` vytvoří `"stringstring"` (brzy si o něm řekneme více)
- odstranit všechny uvozovky a pozorně si prohlédněte výstup Pythonu; věnujte pozornost tomu, kde Python vidí chybu - je to místo, kde chyba skutečně existuje?
- totéž proveďte s některými závorkami;
- změňte některé ze slov `print` na něco jiného, lišícího se pouze v případě (např. `Print`) - co se stane nyní?
- nahraďte některé uvozovky apostrofy; pozorně sledujte, co se stane.

Klíčové poznatky

1. Funkce `print()` je vestavěná funkce. Vytiskne/vypíše zadanou zprávu na obrazovku/do okna konzoly.

2. Vestavěné funkce jsou na rozdíl od uživatelsky definovaných funkcí vždy k dispozici a není nutné je importovat. Python 3.8 obsahuje 69 vestavěných funkcí. Jejich úplný seznam uvedený v abecedním pořadí najdete ve standardní knihovně Pythonu.

3. Chcete-li zavolat funkci (tento proces se nazývá volání funkce nebo volání funkce), musíte použít název funkce následovaný závorkami. Argumenty můžete do funkce předat tak, že je umístíte do závorek. Argumenty musíte oddělit čárkou, např. `print("Hello,", "world!")`. "Prázdná" funkce `print()` vypíše na obrazovku prázdný řádek.

4. Řetězce v jazyce Python se oddělují uvozovkami, např. `"I am a string"` (dvojité uvozovky) nebo `'I am a string, too'` (jednoduché uvozovky).

5. Počítačové programy jsou soubory instrukcí. Instrukce je příkaz, který má po provedení provést určitou úlohu, např. vypsat na obrazovku určitou zprávu.

6. V řetězcích jazyka Python je zpětné lomítko (`\`) speciální znak, který oznamuje, že následující znak má jiný význam, např. `\n` (znak nového řádku) začíná nový výstupní řádek.

7. Poziční argumenty jsou takové, jejichž význam je dán jejich pozicí, např. druhý argument je vypsán za prvním, třetí za druhým atd.

8. Klíčové argumenty jsou takové, jejichž význam není dán jejich umístěním, ale speciálním slovem (klíčovým slovem), které se používá k jejich identifikaci.

9. Parametry `end` a `sep` lze použít pro formátování výstupu funkce print(). Parametr `sep` určuje oddělovač mezi vypisovanými argumenty (např. `print("H", "E", "L", "L", "O", sep="-")`, zatímco parametr `end` určuje, co se má vypsat na konci příkazu `print`.

Odkazy:

Cisco Programming Essentials in Python

Root.cz

ITNetwork.cz

Internet