Úvod do jazyka Python a počítačového programování

(Určeno pro vnitřní potřebu SOUE Plzeň, kopírování bez předchozího souhlasu je zakázáno, zveřejňování na webu je zakázáno)

Modul 2

2. Sekce 6 – Interakce s uživatelem

V této části se naučíte komunikovat s počítačem: seznámíte se s funkcí `input()`, provedete konverzi typů a naučíte se používat řetězcové operátory.

2.6.1 Funkce `input()`

Nyní vás seznámíme se zcela novou funkcí, která se zdá být zrcadlovým odrazem staré dobré funkce `print()`.

Proč? No, funkce `print()` odesílá data na konzolu.

Nová funkce z ní data získává.

Funkce `print()` nemá žádný použitelný výsledek. Smyslem nové funkce je vrátit velmi dobře použitelný výsledek.

Funkce se jmenuje `input()`. Název funkce říká vše.

Funkce `input()` dokáže načíst data zadaná uživatelem a vrátit stejná data běžícímu programu.

Program může s daty manipulovat, čímž se kód stává skutečně interaktivním.

Prakticky všechny programy čtou a zpracovávají data. Program, který nedostává vstupní data od uživatele, je hluchý program.

Podívejte se na náš příklad:

```
print("Tell me anything...")
anything = input()
print("Hmm...", anything, "... Really?")
```

Ukazuje velmi jednoduchý případ použití funkce `input()`.

Poznámka:

Program vyzve uživatele, aby zadal nějaká data z konzoly (nejspíše pomocí klávesnice, i když je možné zadávat data také hlasem nebo obrazem);
funkce `input()` je vyvolána bez argumentů (to je nejjednodušší způsob použití funkce); funkce přepne konzolu do vstupního režimu; uvidíte blikající kurzor a budete moci zadat několik stisků kláves, což zakončíte stiskem klávesy Enter; všechna zadaná data budou odeslána do programu prostřednictvím výsledku funkce;

poznámka: výsledek je třeba přiřadit proměnné; to je klíčové - vynechání tohoto kroku způsobí ztrátu zadaných dat;
poté použijeme funkci `print()` k vypsání získaných dat s několika doplňujícími poznámkami.
Spusťte kód a nechte si ukázat, co pro vás funkce dokáže udělat.

2.6.2 Funkce `input()` s argumentem

Funkce `input()` umí ještě něco jiného: může se uživatele zeptat bez pomoci funkce `print()`.

Náš příklad jsme trochu upravili, podívejte se na kód:

```
anything = input("Tell me anything...")
print("Hmm...", anything, "...Really?")
```

Poznámka:

Funkce `input()` se volá s jedním argumentem - je to řetězec obsahující zprávu;
zpráva se zobrazí na konzoli dříve, než uživatel dostane možnost cokoli zadat;
funkce `input()` pak provede svou práci.
Tato varianta volání funkce `input()` zjednodušuje kód a zpřehledňuje jej.

2.6.3 Výsledek funkce `input()`

Už jsme to říkali, ale je třeba to jednoznačně zopakovat: výsledkem funkce `input()` je řetězec.

Řetězec obsahující všechny znaky, které uživatel zadá z klávesnice. Není to celé číslo ani float.

To znamená, že jej nesmíte použít jako argument žádné aritmetické operace, např. nemůžete tento údaj použít k odmocňování, dělení čímkoli nebo dělení čímkoli.

```
anything = input("Enter a number: ")
something = anything ** 2.0
print(anything, "to the power of 2 is", something)
```

2.6.4 Funkce `input()` - zakázané operace

Podívejte se na následující kus kódu v editoru. Spusťte jej, zadejte libovolné číslo a stiskněte klávesu Enter.

```
# Testing a TypeError message.
anything = input("Enter a number: ")
something = anything ** 2.0
print(anything, "to the power of 2 is", something)
```

Co se stane? Python by vám měl poskytnout následující výstup:

```
Traceback (most recent call last):
File ".main.py", line 4, in <module>
something = anything ** 2.0
```

```
TypeError: unsupported operand type(s) for ** or pow(): 'str' and
'float'
```

Poslední řádek věty vše vysvětluje - pokusili jste se použít operátor `**` na 'str' (řetězec) doplněný o 'float'.

To je zakázáno.

Mělo by to být zřejmé - dokážete předpovědět hodnotu "to be or not to be" zvýšenou na mocninu 2?

Nemůžeme. Python to také neumí.

Dostali jsme se do slepé uličky? Existuje nějaké řešení tohoto problému? Samozřejmě, že existuje.

2.6.5 Přetypování (typové konverze, type casting, type conversions)

Python nabízí dvě jednoduché funkce pro zadání typu dat a řešení tohoto problému - zde jsou: `int()` a `float()`.

Jejich názvy jsou samy o sobě komentující:

Funkce `int()` přijímá jeden argument (např. řetězec: `int(string)`) a snaží se jej převést na celé číslo; pokud se jí to nepodaří, selže i celý program (pro tuto situaci existuje řešení, ale to si ukážeme o něco později);

funkce `float()` přijme jeden argument (např. řetězec: `float(string)`) a pokusí se jej převést na float (zbytek je stejný).

Je to velmi jednoduché a velmi efektivní. Navíc můžete kteroukoli z funkcí vyvolat tak, že jim přímo předáte výsledky funkce `input()`. Není třeba používat žádnou proměnnou jako mezipaměť.

```
anything = float(input("Enter a number: "))
something = anything ** 2.0
print(anything, "to the power of 2 is", something)
```

Vyzkoušejte toto řešení v editoru.

2.6.6 Více o přetypování

Tým složený z trojice `input()-int()-float()` otevírá spoustu nových možností.

Nakonec budete moci psát kompletní programy, které přijímají data ve formě čísel, zpracovávají je a zobrazují výsledky.

Tyto programy budou samozřejmě velmi primitivní a nepříliš použitelné, protože se nemohou rozhodovat, a tudíž nejsou schopny reagovat různě na různé situace.

To však ve skutečnosti není problém; brzy si ukážeme, jak jej překonat.

Náš další příklad odkazuje na dřívější program pro zjištění délky přepony. Spusťme jej a zařiďme, aby dokázal z konzoly vyčíst délky ramen.

```
strana_a = float(input("Vlozte delku strany a: "))
strana_b = float(input("Vlozte delku strany b: "))
prepona = (strana_a **2 + strana_b **2) ** .5
print("Delka prepony je: ", prepona)
```

Program se zeptá uživatele na délky obou ramen, vyhodnotí přeponu a vypíše výsledek. Spusťte jej a zkuste zadat některé záporné hodnoty.

Program bohužel na tuto zjevnou chybu nereaguje. Prozatím tento nedostatek ignorujme. Brzy se k němu vrátíme.

Všimněte si, že v programu, který vidíte v editoru, se proměnná `hypo` používá k jedinému účelu - k uložení vypočtené hodnoty mezi vykonáním sousedního řádku kódu.

Protože funkce `print()` přijímá jako argument výraz, můžete proměnnou z kódu odstranit.

Přesně takto:

```
strana_a = float(input("Vlozte delku strany a: "))
strana_b = float(input("Vlozte delku strany b: "))
print("Delka prepony je: ", (leg_a**2 + leg_b**2) ** .5)
```

Vyzkoušejte program v editoru.

2.6.7 Řetězcové operátory

Je čas vrátit se k těmto dvěma aritmetickým operátorům: `+` a `*`.

Chceme si ukázat, že mají i druhou funkci. Dokážou něco víc než jen sčítat a násobit.

Viděli jsme je v akci, kdy jejich argumenty jsou čísla (plovoucí nebo celá, na tom nezáleží).

Nyní vám ukážeme, že umí pracovat i s řetězci, i když velmi specifickým způsobem.

Znak `+` (plus) se po použití na dva řetězce stane operátorem spojování:

```
string + string
```

Jednoduše spojí (slepí) dva řetězce do jednoho. Jde o operaci konkatenace (concatenate). Stejně jako jeho aritmetický sourozenec může být samozřejmě v jednom výrazu použit více než jednou a v takovém kontextu se chová podle levostranné vazby.

Na rozdíl od svého aritmetického sourozence není operátor spojování komutativní, tj. `"ab" + "ba"` není totéž jako `"ba" + "ab"`.

Nezapomeňte - pokud chcete, aby znak `+` byl konkatenátorem, nikoli sčítačem, musíte zajistit, aby oba jeho argumenty byly řetězce.

Typy zde nemůžete míchat.

Tento jednoduchý program ukazuje znak + v jeho druhém použití:

```
fnam = input("May I have your first name, please? ")
lnam = input("May I have your last name, please? ")
print("Thank you.")
print("\nYour name is " + fnam + " " + lnam + ".")
```

Poznámka: použití + ke spojování řetězců umožňuje konstruovat výstup přesnějším způsobem než čistá funkce `print()`, i když je obohacena o argumenty klíčových slov `end=` a `sep=`.

Spusťte kód a podívejte se, zda výstup odpovídá vašim předpokladům.

Replikace

Znaménko `*` (hvězdička) se při použití na řetězec a číslo (nebo číslo a řetězec, protože v této pozici zůstává komutativní) stává operátorem replikace:

```
string * number
number * string
```

Řetězec se replikuje počtem, který je zadán číslem.

Například:

`"James" * 3` dává `"JamesJamesJames"`.

`3 * "an"` dává `"ananan"`

`5 * "2"` (nebo `"2" * 5`) dává `"22222"` (ne 10!).

Nezapomeňte na

Číslo menší nebo rovno nule vytvoří prázdný řetězec.

Následující jednoduchý program "nakreslí" obdélník, přičemž využije starý operátor (+) v nové roli:

```
print("+" + 10 * "-" + "+")
print(("|" + " " * 10 + "|\n") * 5, end="")
print("+" + 10 * "-" + "+")
```

Všimněte si, jakým způsobem jsme použili závorky v druhém řádku kódu.

Zkuste si procvičit vytváření dalších tvarů nebo vlastních uměleckých děl!

2.6.8 Opět typové konverze

`str()`

Již víte, jak používat funkce `int()` a `float()` k převodu řetězce na číslo.

Tento typ převodu není jednosměrný. Číslo můžete také převést na řetězec, což je mnohem jednodušší a bezpečnější - tento druh operace je možný vždy.

Funkce, která je toho schopna, se nazývá `str()`:

```
str(number)
```

Upřímně řečeno, umí toho mnohem víc než jen převádět čísla na řetězce, ale to může počkat na později.

Opět pravoúhlý trojúhelník

Zde je opět náš program "pravoúhlý trojúhelník":

```
leg_a = float(input("Input first leg length: "))
leg_b = float(input("Input second leg length: "))
print("Hypotenuse length is " + str((leg_a**2 + leg_b**2) ** .5))
```

Trochu jsme ji upravili, abychom vám ukázali, jak funkce `str()` funguje. Díky tomu můžeme funkci `print()` předat celý výsledek jako jeden řetězec a zapomenout na čárky.

Na cestě k programování v jazyce Python jste udělali pořádný krok kupředu.

Znáte již základní datové typy a sadu základních operátorů. Víte, jak uspořádat výstup a jak získat data od uživatele. To jsou velmi silné základy pro modul 3. Než však přejdeme k dalšímu modulu, uděláme si několik cvičení a zrekapitulujeme si vše, co jste se v této části naučili.

2.6.9 LAB Jednoduchý vstup a výstup

Scénář

Vaším úkolem je doplnit kód tak, aby vyhodnotil výsledky čtyř základních aritmetických operací.

Výsledky je třeba vypsat na konzolu.

Kód nemusíte ochraňovat před uživatelem, který chce dělit nulou. To nevadí, zatím si s tím nedělejte starosti.

Otestujte svůj kód - dává očekávané výsledky?

Nebudeme vám ukazovat žádná testovací data - to by bylo příliš jednoduché.

Dokážete si představit, jak řetězec zadaný uživatelem proudí z funkce `input()` do funkce `print()`?

Zkuste si upravený kód spustit. Nezapomeňte zadat platné číslo.

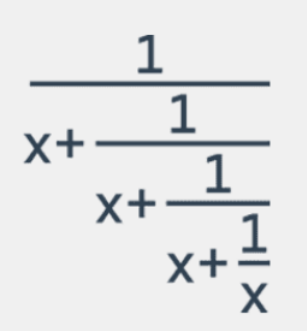
```
# zde vlozte float hodnotu pro promennou a
# zde vlozte float hodnotu pro promennou b

# zde vytisknete vysledek scitani
# zde vytisknete vysledek odcitani
# zde vytisknete vysledek nasobeni
# output the result of division here
print("\nA to je vse!")
```

Vyzkoušejte několik různých hodnot, malé i velké, záporné i kladné. Nula je také dobrý vstup.

2.6.10 Operátory a výrazy LAB

Scénář

Vaším úkolem je doplnit kód tak, abyste vyhodnotili následující výraz:



Výsledek by měl být přiřazen do `y`. Buďte opatrní - sledujte operátory a mějte na paměti jejich priority. Neváhejte použít tolik závorek, kolik potřebujete.

Pro zkrácení výrazu můžete použít další proměnné (ale není to nutné). Svůj kód pečlivě otestujte.

Test Data

Sample input:	Expected output:
1	Output y = 0.6000000000000001
10	Output y = 0.09901951266867294
100	Output y = 0.009999000199950014
-5	Output y = -0.19258202567760344

Vzorový kód:

```
x = float(input("Enter value for x: "))
# Write your code here.
```

```
print("y =", y)
```

2.6.11 Operátory a výrazy LAB – 2

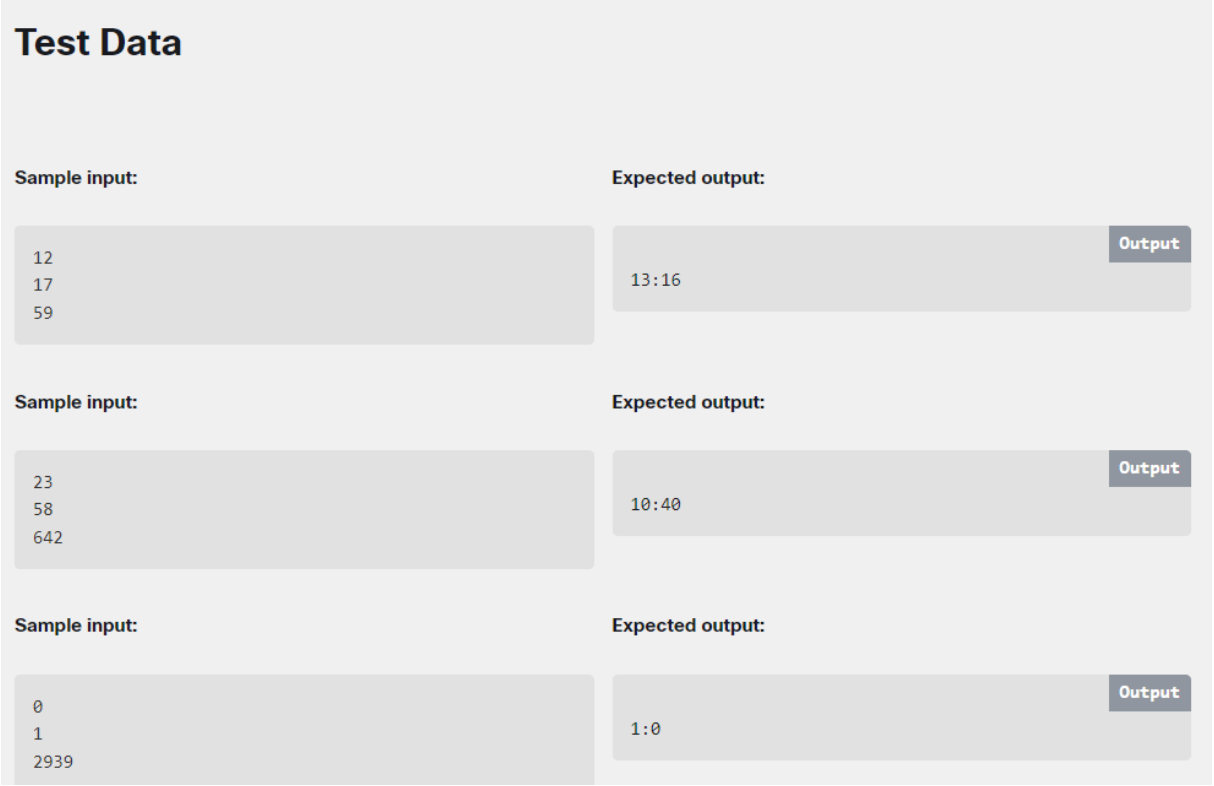
Scénář

Vaším úkolem je připravit jednoduchý kód, který bude schopen vyhodnotit čas konce časového úseku zadaného jako počet minut (může být libovolně velký). Počáteční čas je zadán jako dvojice hodin (0..23) a minut (0..59). Výsledek je třeba vypsat na konzolu.

Pokud například událost začíná ve 12:17 a trvá 59 minut, skončí ve 13:16.

Nedělejte si starosti s případnými nedokonalostmi kódu - nevadí, když přijme neplatný čas - nejdůležitější je, aby kód poskytoval platné výsledky pro platná vstupní data.

Svůj kód pečlivě otestujte. Tip: klíčem k úspěchu může být použití operátoru `%`.



Vzorový kód:

```
hour = int(input("Starting time (hours): "))
mins = int(input("Starting time (minutes): "))
dura = int(input("Event duration (minutes): "))
# Write your code here.
```

2.6.12 Shrnutí sekce

1. Funkce `print()` odesílá data do konzoly, zatímco funkce `input()` získává data z konzoly.

2. Funkce `input()` má nepovinný parametr: řetězec výzvy. Ten umožňuje před uživatelský vstup napsat zprávu, např:

```
name = input("Enter your name: ")
print("Hello, " + name + ". Nice to meet you!")
```

3. Po zavolání funkce `input()` se tok programu zastaví, symbol výzvy stále bliká (vyzývá uživatele k akci, když je konzola přepnuta do vstupního režimu), dokud uživatel nezadá vstup a/nebo nestiskne klávesu Enter.

Poznámka
Funkčnost funkce `input()` v plném rozsahu si můžete vyzkoušet lokálně na svém počítači. Z důvodů optimalizace zdrojů jsme omezili maximální dobu provádění programu v Edube na několik sekund. Přejděte do Sandboxu, zkopírujte a vložte výše uvedený úryvek, spusťte program a nedělejte nic - počkejte několik sekund a sledujte, co se stane. Váš program by se měl po krátké chvíli automaticky zastavit. Nyní otevřete IDLE a spusťte stejný program tam - vidíte rozdíl?

Tip: výše zmíněnou vlastnost funkce `input()` lze použít k vyzvání uživatele k ukončení programu. Podívejte se na níže uvedený kód:

```
name = input("Enter your name: ")
print("Hello, " + name + ". Nice to meet you!")

print("\nPress Enter to end the program.")
input()
print("THE END.")
```

4. Výsledkem funkce `input()` je řetězec. Řetězce můžete k sobě přidávat pomocí operátoru spojování (+). Podívejte se na tento kód:

```
name = input("Enter your name: ")
print("Hello, " + name + ". Nice to meet you!")
print("\nPress Enter to end the program.")
input()
print("THE END.")
```

5. Řetězce můžete také násobit (* - replikace), např.:

```
my_input = input("Enter something: ") # Example input: hello
print(my_input * 3) # Expected output: hellohellohello
```

2.6.13 Kvíz sekce

Otázka 1: Jaký je výstup následujícího úryvku kódu?

```
x = int(input("Enter a number: ")) # The user enters 2
print(x * "5")
```

Otázka 2: Jaký je očekávaný výstup následujícího úryvku kódu?

```
x = input("Enter a number: ") # The user enters 2
print(type(x))
```

Odkazy:

Cisco Programming Essentials in Python

Root.cz

Internet