

1. Programování 1 pro matematiky

2. cvičení, 6-10-2022

tags: Programovani 1 2022, čtvrtek 1, čtvrtek 2

Obsah:

- 0. Farní oznamy
- 1. Opakování
- 2. Příkaz `if`
- 3. Příkaz `while`
- 4. Programujeme...

Farní oznamy

1. **Materiály k přednáškám** najdete v GitHub repozitáři <https://github.com/PKvasnick/Programovani-1>. Najdete tam také kód ke cvičením.
 - Soubory si můžete číst přímo na GitHubu. Pokud si chcete stáhnout nebo zkopírovat kód, přepněte do *Raw* zobrazení (aby se vám nezkopírovaly čísla řádků a pod.), Ctrl-A + Ctrl-C.
 - Windows: Nainstalujte si aplikaci GitHub Desktop a naklonujte si celý repozitář do svého počítače: Zelené tlačítko `Code`, z nabídky `open with GitHub Desktop`.
 - Pokud se v nějakém okamžiku neobejdete bez zřízení konta na GitHubu, zřídte si jej.
2. **Domácí úkoly** Doiterovali jsme se k docela velké úspěšnosti u jediného úkolu, který jsem zadával na předchozím cvičení. Více v osobitné části.

Pokud ještě nemáte správné řešení, vězte, že dostanete ještě nějaké body i když ho odevzdáte do příští středy.

3. **Každému vše chodí?** - přišel takovýto e-mail, pokud máte problémy, prosím hlašte se:

Zdravím,

Někteří z nás na cvičeních čelili problémům studentů s nemožností se přihlásit do počítačů v Impactu. Závěry z vyšetřování příčin a řešení posílám níže:

\\1. Kdo není studentem MFF, tak se nemůže přihlásit (systém uznává pouze jmenný login a ten studenti jiných fakult prý nemají). Řešením je zřídit jim lokální účet na Malé Straně. Jana zjistila, že je možné napsat email na simunek@sisal.mff.cuni.cz jméno s číslem osoby na kartičce UK (a pro úplnost i email) a Dr. Šimůnek jim to zařídí.

\\2. Studenti MFF, kteří mají účet na Malé Straně, se musí mít na pozoru. Tento účet má prioritu před přihlašováním přes CAS. Takže pro přihlášení na počítače v Impactu musí! použít login a heslo z Malé Strany. Toto platí i pro toho, kdo si časem účet na Malé Straně ještě udělá.

\\3. Pro přihlášení pomocí CAS nefunguje číslo osoby. Je nutné zadat login, který se skládá z části příjmení, jména, čísla a nějakých písmen, typu roskb5am. Pro přihlášení je prý nutné použít malá písmena.

\\4. Studenti mohou mít ještě problémy s přihlášením, páč na počítači může být nastavená anglická klávesnice a ne česká.

Pokud máte ještě jiné postřehy, neváhejte je sdílet.

Zatím,

Beda Roskovec

Opakování

- ✓ Základní instalace Pythonu
- ✓ Čísla a řetězce, aritmetické a logické operace
- ✓ **Domácí úkol:** Konverze `int()`

```
1 a = input()
2 b = input()
3 print(a, b, a>b)
4
5 ===== RESTART: C:\Users\kvasn\Dropbox\Python_MFF\sk.py
6 =====
7 2
8 13
9 2 13 True
```

```
1 a = int(input())
2 b = int(input())
3 print(a, b, a>b)
4
5 ===== RESTART: C:\Users\kvasn\Dropbox\Python_MFF\sk.py
6 =====
7 2
8 13
9 2 13 False
```

Nejspíš vás nepřekvapí, že také existuje `float()`, `str()` a `bool()`

```
1 In [3]: int(4.9)
2 Out[3]: 4
3
4 In[4]: int("Petr")
5 Traceback (most recent call last):
6   File "<pyshell#72>", line 1, in <module>
7     int("Petr")
8 ValueError: invalid literal for int() with base 10: 'Petr'
9
10 In [5]: round(4.9,0)
```

```

11 Out[5]: 5.0
12
13 In [6]: float(5)
14 Out[6]: 5.0
15
16 In [7]: bool(0.5)
17 Out[7]: True
18
19 In [8]: bool(-1.0)
20 Out[8]: True
21
22 In [9]: bool(0.0)
23 Out[9]: False
24
25 In [10]: str(4.6)
26 Out[10]: '4.6'
27
28 In [11]: str(True)
29 Out[11]: 'True'
30
31 In [12]: str(False)
32 Out[12]: 'False'
33
34

```

- ☐ Operátory `+, -, *, /, **, //, %, ==, and, or, not`
- ☐ Přiřazení `=` a přiřazení s operací `+=, -=, *=, /=`, ale také třeba `%=` - operátor *vymodulení*, s kterým se dnes setkáme.
- ☐ Matematické funkce z balíku *math*, `import math` a pak `math.*`, např. `math.sin()`.
- ☐ Funkce pro čtení řetězce ze standardního vstupu `input(výzva)` a funkce pro tisk do standardního výstupu `print(objekt1, objekt2, ...)`

Print podrobněji:

```

1 print(1,2,3); print(4,5,6)
2 1 2 3
3 4 5 6

```

Konverze do řetězcové reprezentace, položky oddělené mezerami, na konci znak nového řádku.

```

1 print(1, 2, 3, sep = "-", end = "!!!\n")

```

Formátování výstupu:

```

1 jmeno = "Petr"
2 vaha = 100
3 print(jmeno, "váží", vaha, "kilogramů")
4 print(f"{jmeno} váží {vaha} kilogramů")

```

- ☐ Podmíněný příkaz

```
1 if podmínka:
2     příkazy
```

☐ Příkaz cyklu

```
1 while podmínka:
2     příkazy
```

kde *příkazy* mohou být příkazy přiřazení, volání funkce, další podmíněné příkazy nebo příkazy cyklu, a dnes se naučíme, že také příkazy `pass` (nedělej nic), `break` (opuštění cyklu) a `continue` (přechod na další iteraci cyklu).

Příkaz `if`

Úplnější syntaxe příkazu `if`:

```
1 if podmínka:
2     příkazy
3 else:           # volitelně
4     příkazy
```

Větev `else` je nepovinná; když chceme vynechat příkazy ve větvi `if`, musíme použít prázdný příkaz `pass`.

Větve `elif`: V případě řetězcích příkazů `if` můžeme namísto konstrukce

```
1 if podmínka1:
2     příkazy
3 else:
4     if podmínka2:
5         příkazy
6     else:
7         příkazy
8
```

psát

```
1 if podmínka1:
2     příkazy
3 elif podmínka2:
4     příkazy
5 else:
6     příkazy
```

což je o něco přehlednější - hlavně díky plochému (nerostoucímu) odsazení.

Příkaz `while`

```
1 while podmínka:
2     příkazy
```

Příkazy pro kontrolu běhu cyklu:

`break` - v tomto místě opustit cyklus a pokračovat příkazem, následujícím za cyklem

`continue` - v tomto místě přejít na další iteraci cyklu (tedy na testování podmínky)

Nekonečný cyklus: podmínka stále platí, a o ukončení cyklu rozhodneme v těle za použití příkazu `break`:

```
1 while True:
2     příkazy
3     if podmínka:
4         break
```

Příkaz `while` má také volitelnou větev `else`. Příkazy v této větvi se vykonají, pokud cyklus řádně skončí (tedy ne v případě opuštění cyklu příkazem `break`).

```
1 while podmínka:
2     příkazy1
3 else:
4     příkazy2
```

Příklady

Test prvočísel

Chceme otestovat, zda je číslo n ze vstupu prvočíslo.

Metoda: U všech čísel $d < n$ prověřím, zda jsou děliteli n .

```
1 #!/usr/bin/env python3
2
3 # Otestuje, zda číslo je prvočíslem
4
5 n = int(input())
6 d = 2
7 mam_delitele = False
8
9 while d < n:
10     if n%d == 0:
11         print("Číslo", n, "je dělitelné", d)
12         mam_delitele = True
13         break
14     d += 1
15
16 if not mam_delitele:
17     print("Číslo", n, "je prvočíslo")
```

To není nijak zvlášť efektivní metoda, ale to nám nevadí, my jsme celí rádi, že umíme napsat něco, co v zásadě funguje.

Pojďme opatrně vylepšovat. Zásadní vylepšení kódu by bylo, kdybychom "nahý" cyklus `while` uměli celý zapouzdřit do jediného příkazu.

🤓 Pokročilé kolegy poprosím o tvar onoho jediného příkazu.

Asi první věc, která nám vadí, je stavová proměnná `max_delitele`. A té se v prvním kroku zbavíme za použití větve `else`:

```
1  #!/usr/bin/env python3
2
3  # Otestuje, zda číslo je prvočíslem (2. pokus)
4
5  n = int(input())
6  d = 2
7
8  while d < n:
9      if n%d == 0:
10         print("Číslo", n, "je dělitelné", d)
11         break
12     d += 1
13 else:
14     print("Číslo", n, "je prvočíslo")
```

Jak bychom mohli dál vylepšit náš test?

Popřemýšlíme, a zatím vymyslíme, jak bychom vypsalí všechna prvočísla menší nebo rovná n . Nejjednodušší metoda bude projít všechna čísla od 2 do n , u každého rozhodnout, zda je prvočíslem, a jestli ano, vypsat ho.

```
1  #!/usr/bin/env python3
2  # vypíše všechna prvočísla od 1 do n
3
4  n = int(input())
5
6  x = 2
7  while x <= n:
8      d = 2
9      while d < x:
10         if x%d == 0:
11             break
12         d += 1
13     else:
14         print(x)
15
16     x += 1
17
```

Optimalizace je v tomto případě ještě více nasnadě, jenomže si zatím neumíme pamatovat věci - například všechna prvočísla, které jsme dosud našli.

🤓 Pokročilé kolegy poprosím o optimalizovaný algoritmus, např. Erastovenovo síto.

Součet posloupnosti čísel

```
1  #!/usr/bin/env python3
2
3  # Načteme ze vstupu posloupnost čísel, ukončenou -1.
4  # Vypíšeme jejich součet.
5
6  s = 0
7  while True:
8      n = int(input())
9      if n == -1:
10         break
11     s += n
12 print(s)
```

Proč nemůžeme na konci jenom stisknout Enter a nezadat nic?

🧐 Pokročilé kolegy poprosím

- o variantu se stiskem Enter
- a pro vypsání aritmetického průměru a standardní odchylky.

Domácí úkoly na příští týden:

- Spočíst a vypsát počet cifer zadaného celého čísla
- Vypsát zadané číslo jako součin prvočinitelů
- Vypočtete Eulerovu funkci (*Euler's totient function*) $\phi(n)$, která je rovna počtu s n nesoudělných (*relatively coprime*) čísel menších než n .