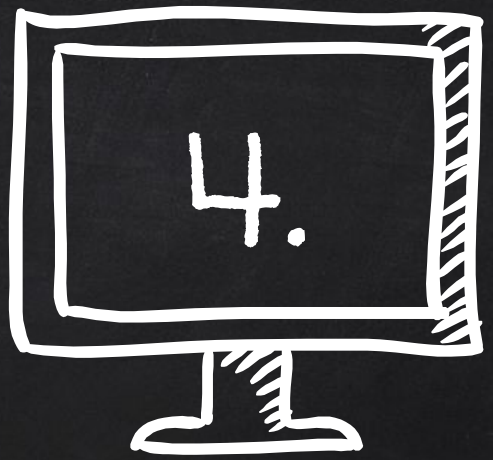
IB111 Úvod do programovaní skrze Python



Autor: Slavomír Krupa,
Text inšpirovaný: Valdemarom Švábenským

Bad habits

```
if this_is_bad:
    a *= 2
    print(a,"#")
else:
    a *= 3
    print(a, "#")
```

```
if this_is_bad:
    a *= 2
else:
    a *= 3
print(a,"#")
```

Bad habits

```
a = 1
for i in range(1,5):
    do_something(a)
    a += 1
```

```
for i in range(1,5):
    do_something(i)
```

I,J,K,L,...

Bad habits

```
for _ in range(5):
    print(" ",end="")
```

```
print(5 * " ",end="")
```

Bad habits

```
def do_something(x):
    x += 13
    return x
    print(x)
```

```
def do_something(x):
    x += 13
    print(x)
    return x
```

Bad habits

```
if random() <= 0.56:
    return True
else:
    return False
```

```
return random() <= 0.56
```


IMPORT

Import

- V ktorom bloku je viditelná aká ~~premenná~~ funkcia?
- Globálne ~~premenné~~ funkcie viditeľné všade

```
from turtle import *
t = Turtle()
t.forward(100)
t.right(90)
t.left(90)
```

Import

- ★ Štandardne vidíme len globálne funkcie a funkcie z nášeho modulu
- ★ Potrebujeme pracovať s inými funkciami

```
>>sqrt(100)
```

3 SPÔSOBY

1. Generický import

```
import math
```

2. Funkčný import

```
from math import sqrt
```

3. Univerzálny import

```
from math import *
```

Generický import

```
import math
```

```
math.sqrt(5)
math.ceil(-4.2)
sqrt(0)
ceil(-4.2)
```

Funkčný import

```
from math import sqrt
```

```
math.sqrt(5)
math.ceil(-4.2)
sqrt(0)
ceil(-4.2)
```

Funkčný import

```
from math import sqrt
from math import ceil
```

```
math.sqrt(5)
math.ceil(-4.2)
sqrt(0)
ceil(-4.2)
```

Univerzálny import

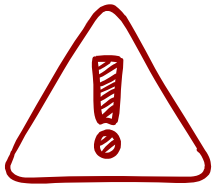
```
from math import *
```

```
math.sqrt(5)
math.ceil(-4.2)
sqrt(4)
ceil(-4.2)
```

Univerzálny import

```
from devil import *
```

```
math.sqrt(5)
math.ceil(-4.2)
sqrt(4)          == 3
ceil(-4.2)       == 6
```



NÁHODA

Náhodné čísla

- Rovnomerne rozložené dáta z nejakého intervalu
- Generátor náhodných čísel vytvára také čísla, ktoré nevieme predpovedať na základe pozorovania predošlých hodnôt ⇒ Generátor má vysokú entropiu (mieru neurčitosti)
  - Generátor s malou entropiou : 4, 4, 4, 4, 4, 4, 4, . . .
  - Kocka: 6, 6, 2, 1, 4, 1, 6, . . .
- Príklady použitia:
  - Simulácia náhody v hrách (hod kockou)
  - Matematické simulácie (Metóda Monte Carlo)

Pseudonáhodné čísla

- Generované zo semienka (seed), napr. zo systémového času/ zvukovej karty/pohybu myši/…
- Generované deterministickým algoritmom (= pre rovnaké semienko poskytne generátor vždy rovnaký výstup)
- V Pythone algoritmus `Mersenne twister`
- Nazývame ich náhodné; v skutočnosti sú pseudonáhodné
- To niekedy stačí, ale inokedy nie (napr. kryptografia)
- Skutočná náhodnosť - musí mať zdroj mimo počítač

Náhoda v pythone

import!

```
...

a = randint(1,10)
b = random()
```

A. int v rozsahu [1,10]
B. float z rosahu [0,1)

Pseudonáhoda v pythone

```
...

seed(8)
a = randint(1,10)
b = random()
```

```
a = 4
b =
0.3704118725828456
```

Pseudonáhoda v pythone

```
...

seed(8)
a = randint(1,10)
b = random()
```

```
a = 4
b =
0.3704118725828456
```

DÚ

Úloha 1

- Naprogramujte funkciu, ktorá bude simulovať správanie kocky (bude náhodne generovať čísla z rosahu 1-6)
- `dice()`

```
from random import randint
from random import random
```

Úloha 2

- Napíšte funkciu simulujúcu hod “pokazenou” mincou. Pokazená minca má 56% pravdepodobnosť, že padne hlava.
  - Funkcia nech vráti True v prípade, že padne hlava - ináč False.
- `flip_broken_coin()`

Úloha 3

- Naprogramujte funkciu, ktorá bude opakovane hádzať kockou pokiaľ bude padať liché (nepárne) číslo.
  - použite funkciu z úlohy 1
- `roll_till_even()`

Úloha 4

- Napíšte procedúru, ktorá spraví `count` hodov kockou a následne vypíše minimum/maximum a priemernú hodnotu.
- `roll_stats(count)`

OPILEC

Zadanie

- Opilec je v polke cesty medzi domom a krčmou (hospodou). Každý krok urobí náhodne jedným smerom. Napíšte funkciu, ktorá bude simulovať opilcov pohyb.
- Parametry budú vzdialenosť medzi domovom a krčmou (hospodou) a počet rozhodnutí do opilcového zaspatia (usnutí)(tj. max dlžka simulácie).
- Simulace skončí:
  - Keď opilec príde domov
  - alebo do hospody,
  - prípadne po vyčerpání počtu krokov.

Výstup

```
home . . . . . * . . . . pub
home . . . . * . . . . . pub
home . . . * . . . . . . pub
home . . . . * . . . . . pub
home . . . * . . . . . . pub
home . . * . . . . . . . pub
home . * . . . . . . . . pub
home . . * . . . . . . . pub
home . . . * . . . . . . pub
home . . . . . * . . . . pub
home . . . . . * . . . pub
home . . . . . * . . . . pub
home . . . . . * . . . pub
home . . . . . * . . pub
home . . . . . * . pub
home . . . . . * pub
home . . . . . * . pub
home . . . . . * pub
home . . . . . . pub
   Ended in the pub again!
```

Step by step 1

- Napíšte procedúru (`output`), ktorá vypíše pozíciu opilca.
- Bude mať 2 parametre
  - `distance` - vzdialenosť medzi domom a krčmou
  - `position`- pozícia opilca

Step by step 2

- Napíšte procedúru `drunk_man_simulation`
- Bude mať 2 parametre
  - `distance` - vzdialenosť medzi domom a krčmou
  - `steps` - počet krokov simulácie
- Overte dátové typy a rozsahy pre vstupné parametre
- vypíšte počiatočnú pozíciu opilca

Step by step 3

- Rozšírte procedúru `drunk_man_simulation`
- Nech sa vykoná najviac `steps` opakovaní kroku opilca
- Nech je cyklus ukončený predčasne v prípade, že opilec dorazí domov/ do krčmy
- Krok - simulácia pohybu - výpis stavu
- Skúste napísať 2 verzie:
  - For cyklus + Break
  - While cyklus

Step by step 4

- Rozšírte procedúru `drunk_man_simulation`
  - o parameter `outputEnabled` (bool) - bude určovať či sa má vypisovať stav opilca
  - o návratovú hodnotu
    - 1 opilec príde domov
    - 0 opilec usne
    - - opilec skončí v hospode

DONE...

...ALMOST

Simulácia

- Napíšte funkciu `drunk_man_analysis` s tromi parametrami:
  - `count` - koľkokrát spustí funkciu s opilcom
  - `distance` - vzdialenosť medzi krčmou a domom
  - `steps` - maximálny počet spustení simulácie
- ,ktorá vypíše štatistky pre všetky možnosti ukončenia.
- Poznámka: pri simulácii nevypisujte stav.

Úloha 5*

★ Napíšte procedúru, ktorá spraví `count` hodov kockou a následne vypíše koľkokrát, ktoré číslo padlo.
- použite funkciu z úlohy 1
- Budete potrebovať zoznamy: https://www.fi.muni.cz/IB111/sbirka/05-retezce_a_seznamy.html

★ `roll_stats(count)`

Zbierka*

- https://www.fi.muni.cz/IB111/sbirka/04-nahodna_cisla.html
- 4.2.3
- 4.2.4
- 4.2.5
- 4.2.6

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash