IB113 Úvod do programování a algoritmizace

Cvičení 1 – organizace předmětu, základní konstrukce, seznámení s IDE a želví grafika

Jaromír Plhák

Cíle předmětu

- Motivovat k programování
- Naučit se řešit problémy a algoritmicky myslet
- Vysvětlit všeobecné principy programování
  - Jak? Psát a číst kód!
- Ilustrovat základní konstrukce v jazyku Python
  - Ne detaily jazyka

Výhody a nevýhody Pythonu

+ Jednoduchá syntaxe
+ Minimum kódu (téměř pseudokód)
+ Dobrá čitelnost kódu
+ Rozsáhlá standardní knihovna
+ Nezávislost na platformě

− Nemožnost kontroly datových typů při překladu
− Oproti staticky a explicitně typovaným jazykům náchylnost k chybám programátora

Organizace předmětu

- Vše potřebné v interaktivní osnově
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/index.qwarp
- Absence budu evidovat v ISu
  - Pokud dojde k nějaké nesrovnalosti, informujte mne prosím mailem
- Domácí úlohy budou zadány v pondělí (případně úterý)
  - Odevzdání do následujícího čtvrtku
  - Zpětná vazba do poznámkového bloku
  - Nutno získat z každého úkolu alespoň jeden bod
- Zápočtový test
  - Poslední týden v semestru v rámci cvičení
  - Jedna možnost opravy – termín kolem 18. prosince

Základní konstrukce

- print ("Hello world !")
- Vypsat číslo 1
  - print ("1")
- Vypsat čísla od 1 do 3
  - print ("1")
  - print ("2")
  - print ("3")
- Vypsat čísla od 1 do 1000?

Základní konstrukce – proměnné (1)

- Udržují hodnotu
- Udržovaná hodnota se může měnit – proto proměnné
- Typy:
  - Číselné: int, float, ...
  - Pravdivostní hodnota (bool)
  - Řetězec (string)
  - Seznam / pole
  - Slovník
  - ...

Základní konstrukce – proměnné (2)

- Přiřazení =
  - x = 3 znamená „přiřaď do x hodnotu 3“
- Test na rovnost ==
  - x == 3 znamená „otestuj zda v x je hodnota 3
- Častou chybou je záměna = a ==
- Operace
  - +, -, *, /, and, or
  - Umocňování pomocí **
  - Dělení se zbytkem % (modulo)
  - Celočíselné dělení: //
  - Zkrácený zápis „y += 5“ znamená „y = y + 5“

Základní konstrukce – podmíněný příkaz

```
if <podmínka>:
        příkaz1
else:
        příkaz2
```

- Podle toho, zda platí podmínka, se provede jedna z větví
- Podmínka – typicky výraz nad proměnnými
- else větev nepovinná
- Vícenásobné větvení: if - elif - ... - else

Základní konstrukce – cyklus (1)

- Opakované provádění sekvence příkazů
- Známý počet opakování cyklu
- Příkaz for

```
for x in range(n):
    příkazy
```

- range(n) – interval od 0 do n-1 (tj. n opakování)

Základní konstrukce – cyklus (2)

- Neznámý počet opakování cyklu
- Příkaz while
- Opakuj dokud není splněna podmínka

```
while <podmínka>:
        příkazy

f = 10
while f > 0:
        f -= 2
        print(f)
```

Úkoly (1)

- Který příkaz uloží do proměnné y hodnotu 3 v případě, že proměnná x je rovna hodnotě 5?

```
if x = 5:
        y == 3

if x == 5:
        y = 3
```

Úkoly (2)

- Jak tedy vypíši čísla od 1 do 1000?

```
for i in range(10):
        print(i + 1)
```

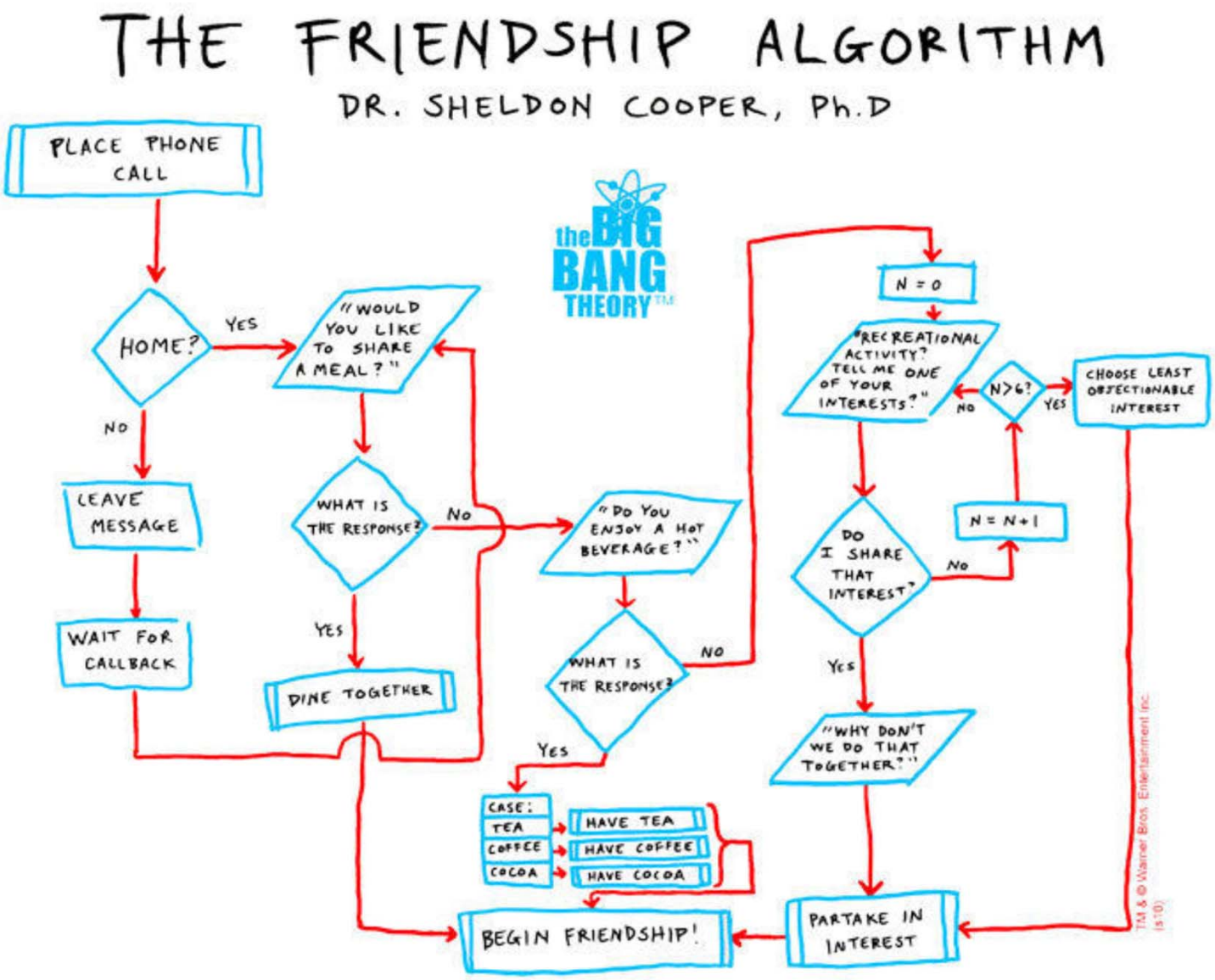
- Jak upravím tento cyklus tak, aby mi vypsal jen čísla dělitelná pěti?

```
for i in range(100):
        if (i + 1) % 5 == 0:
                print(i + 1)
```

- Jiné nápady?

Algoritmus a program

- Algoritmus: konečná postupnost přesně definovaných instrukcí pro splnění určité úlohy
  - [MCCONNELL, Steve: *Code Complete: A Practical Handbook of Software Construction*. Prvé vydanie. Microsoft Press, 1993. 857 strán. ISBN 1-55615-484-4.]
  - Reprezentace: slovně, graficky, pseudokódem
  - Např.: Vypiš na obrazovku číslo jedna!
- Program: přepis algoritmu do programovacího jazyka
  - Reprezentace: soubor s kódem
  - Např.: print("1")
- Cíl předmětu: vymýšlet algoritmy pro řešení problémů a zapisovat algoritmy ve formě programů

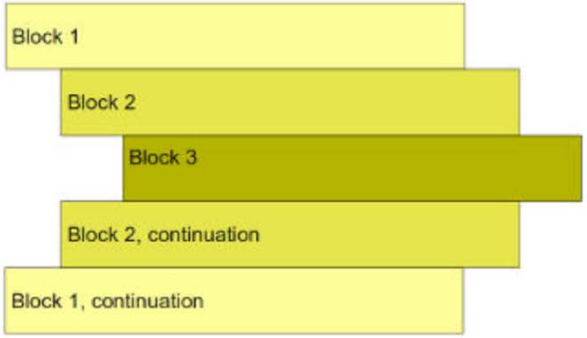


http://www.funnyjunk.com/funny_pictures/478395/The/

```
def make_friends (f):
        phone_call (f)
        if f. is_home :
                accepts = invite_for_meal (f)
                if accepts :
                        have_dinner (f)
                        begin_friendship (f)
                else :
                        …
        …
```

Syntaxe Pythonu (1)

- Case sensitive (ne cASe INSensiTIvE)
- Konec příkazu: konec řádku (ne ;)
- Řádkový komentář: #
- Blokový komentář: """
- Bloky kódu: odsazením (ne {} ani begin. . . end)




```
from math import sqrt
n = input("Maximal Number? ")
n = int(n)+1
for a in range(1,n):
    for b in range(a,n):
        c_square = a**2 + b**2
        c = int(sqrt(c_square))
        if ((c_square - c**2) == 0):
            print(a, b, c)
```

Syntaxe Pythonu (2)

- Méně znaků, přirozená čitelnost
  - Není nutné používat všude závorky
- Dobrý zvyk: po dvojtečce nový řádek a 4 mezery
  - Nastavení znaku Tab v IDE

Seznámení s IDE - PyCharm

- Vytvořit projekt
- Vytvořit nový soubor (“Python file”)
- Nastavit klávesové zkratky
  - CTRL + ALT + S
  - Keymap
    - NetBeans
  - Editor -> Code Style -> Python -> Tab size (nastavit na 4)
- Napsat program vypisující čísla od 1 do 100, která jsou dělitelná pěti
- Spustit program
  - Nejdříve F5, poté F6

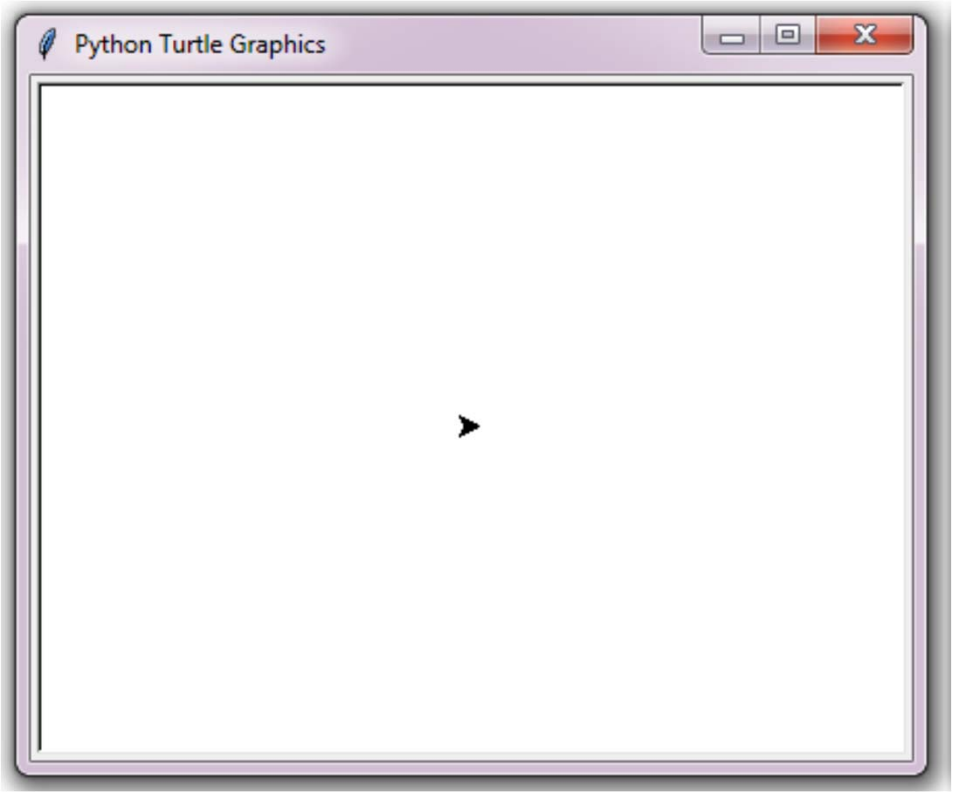
Želví grafika

- **from turtle import Turtle, done**

- Zpřístupní přikazy pro ovladaní želvy
- Inicializace želvy

```
name = Turtle()
name.reset()
done()
```

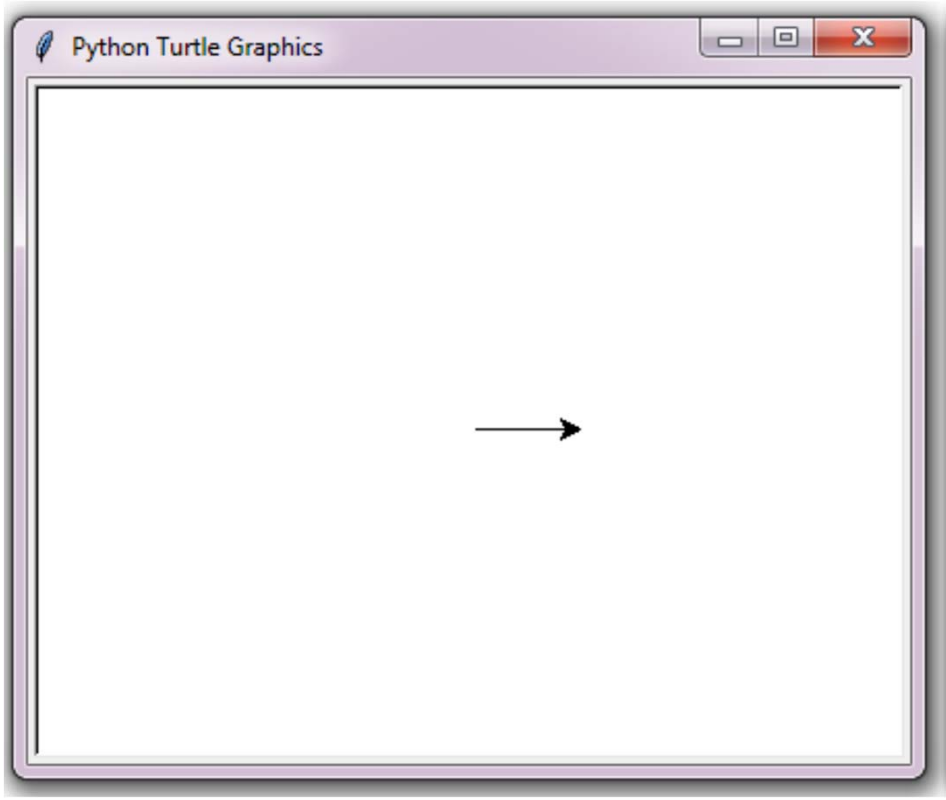
- Nastaví želvu na počáteční pozici
- Funkce done
  - Volá se vždy na konci programu
  - Aby zůstalo okno s grafikou otevřené
- Váš skript se **nesmí** jmenovat turtle.py!



Želví grafika - pohyb

- name.forward(x) – pohyb o x pixelů v daném směru

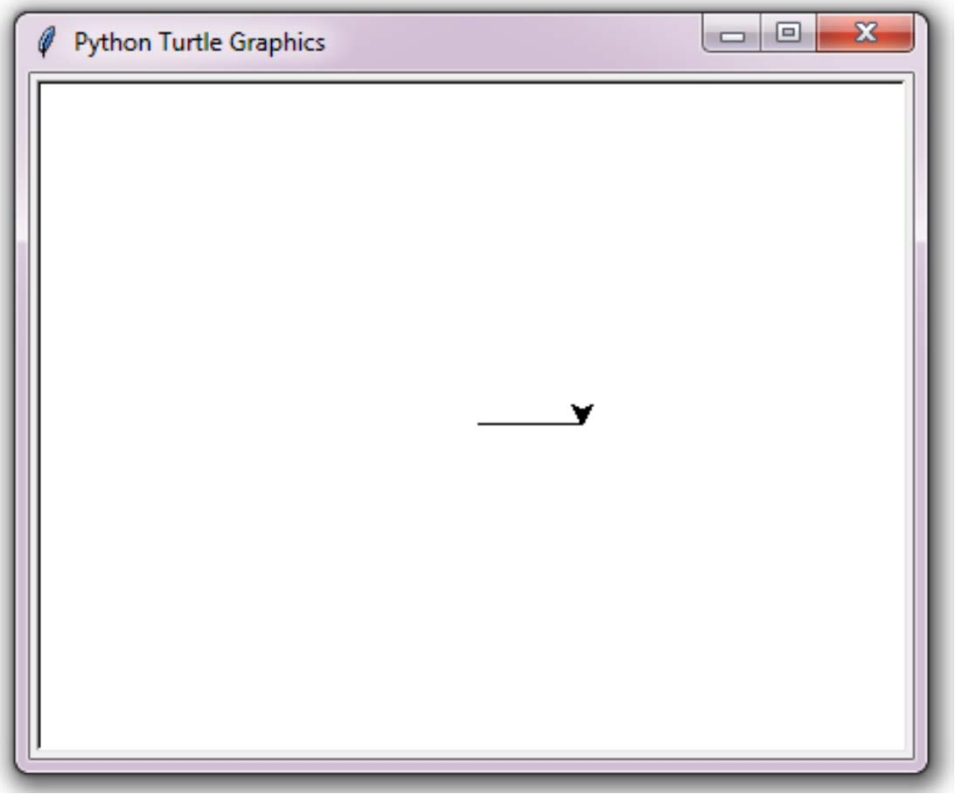
barbara.forward(50)



Želví grafika - otočení

- name.right(x) - rotace o x stupňů doprava
- name.left(x) - rotace o x stupňů doleva

barbara.right(90)

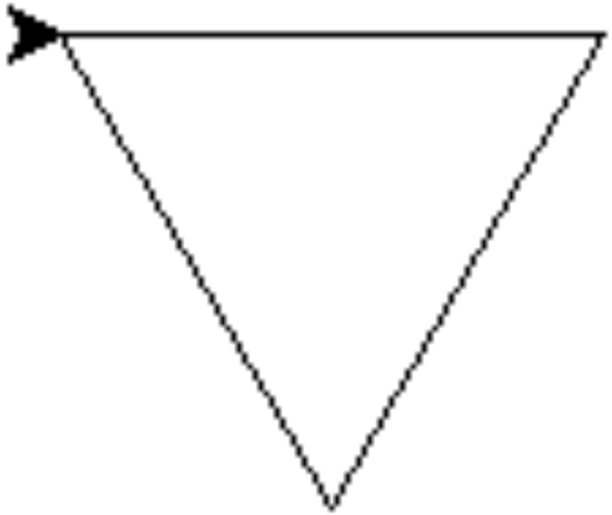


Želví grafika – možnosti

- Želva má
  - Pozici (souřadnice)
  - Otočení
  - Vlastnosti psaní (barva, šířka čáry, ...)
- speed(10) - nejrychlejší postupné vykreslování
  - Lze snížit rychlost

Želví grafika – trojúhelník poprvé

```
barbara.forward(100)
barbara.right(120)
barbara.forward(100)
barbara.right(120)
barbara.forward(100)
barbara.right(120)
```



Želví grafika – trojúhelník ve funkci

- Příkazy nechceme psat opakovaně při tvorbě více trojúhelníků
- Funkce
  - Kouzelné slůvko def
  - A zanoření

```
def triangle(my_turtle):
        my_turtle.forward(100)
        my_turtle.right(120)
        my_turtle.forward(100)
        my_turtle.right(120)
        my_turtle.forward(100)
        my_turtle.right(120)
triangle(barbara)
```

Želví grafika – trojúhelník s cyklem

- Příkaz for i in range(3): způsobí 3 opakování

```
def triangle(my_turtle):
        for i in range(3):
                my_turtle.forward(100)
                my_turtle.right(120)
triangle(barbara)
```

Želví grafika – délky hrany trojúhelníku nastavitelná parametrem

```
def triangle(my_turtle, length):
        for i in range(3):
                my_turtle.forward(length)
                my_turtle.right(120)

triangle(barbara, 250)
```

A co ted?