

Python, Thonny a

E 3011

Jan Böhm

RECETOX

February 26, 2024

Co nás dnes čeká

1 Python 🐍

2 Thonny

3

1	2
3	4

4 Proměnné

5 Funkce

6 🐢



Proč Python?

- 1 **vysokoúrovňový** – zápis se podobá spíše lidskému myšlení, než strojovému
- 2 **interpretovaný** – snadno se dělají malé úpravy
- 3 **oblíbený** – existuje aktivní komunita uživatelů a spousta knihoven
- 4 **dostupný** – můžete ho používat na Windows, Linux, macOS, iOS, Androidu, Raspberry Pi...
- 5 **používaný** – objevuje se pravidelně na předních příčkách požadovaných jazyků

Python je skvělý vstupní jazyk do světa programování.

Zen of Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one – and preferably only one – obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea – let's do more of those!

Co nás dnes čeká

1 Python 

2 Thonny

3 

4 Proměnné

5 Funkce

6 

Co je Thonny?

Thonny je IDE (Integrated Development Environment, česky vývojové prostředí) pro Python. Jaké to má výhody?

- Nemusíme psát kód do poznámkového bloku
- Zvýrazňování syntaxe
- Dokončování výrazů
- Debugging
- ...

Python – demo

Ve studijních materiálech je `demo.py`. Podíváme se, jak vypadá kód v jazyce Python a jak nám pomůže Thonny.

- 1 Spusťte kód pomocí `F5`
- 2 Spusťte kód pomocí `ctrl` + `F5` a následně se posouvejte pomocí `F6`
- 3 Co dělá `#`?
- 4 Co dělá `tempF = float(tempF)`?
- 5 Co dělá `%`?
- 6 Co napíše `isLeapYear("2024")`?

Co nás dnes čeká

1 Python 

2 Thonny

3 

4 Proměnné

5 Funkce

6 

Math – demo

Bez výpočtů se neobejdeme. Ukážeme si, jak v Pythonu zapsat:

- sčítání, odčítání, násobení, dělení
- umocňování
- celočíselné dělení a modulo
- zaokrouhlování
- konstanty π a e
- goniometrické funkce
- logaritmy

```
1 import math # module math contains mathematical functions such
   as trigonometrics
2
3 help(math) # help for math
4 help(math.sin) # help for function sin from library math
```

Co nás dnes čeká

1 Python 

2 Thonny

3 

4 **Proměnné**

5 Funkce

6 

Co je to proměnná?

Proměnná je pojmenovaná hodnota.

Co je to proměnná?

Proměnná je pojmenovaná hodnota.

Vytvoření proměnné

Pro vytvoření proměnné (přiřazení hodnoty) používáme =.

Co je to proměnná?

Proměnná je pojmenovaná hodnota.

Vytvoření proměnné

Pro vytvoření proměnné (přiřazení hodnoty) používáme =.

```
1 year = 2023
2 print(year)
```

Změna hodnoty proměnné?

Hodnotu v proměnné můžeme měnit opět pomocí přiřazení =. Při této změně můžeme využít hodnotu této i ostatních proměnných.

```
1 year = 2023
2 year = year + 1
3 year += 1 # same as above, pythonic
4 print(year)
5
6 a = 3
7 b = 4
8 c = (a**2 + b**2) ** (1/2)
9 print(c)
```

Základní typy proměnných?

Dnes si představíme 4 základní a jeden speciální typ proměnných:

- `int` – integer, celé číslo
- `float` – necelé číslo
- `str` – string, textový řetězec
- `bool` – boolean (`True`, `False`)
- `None` – chybějící hodnota, speciální


```
1 UCO = "408849" # quotes means string
2 print(UCO)
3 print(type(UCO))
4
5 UCO = int(UCO) # convert to integer
6 print(UCO)
7 print(type(UCO))
8
9 UCO = float(UCO) # convert to float
10 print(UCO)
11 print(type(UCO))
12
13 UCO = bool(UCO) # convert to boolean
14 print(UCO)
15 print(type(UCO))
```

Co nás dnes čeká

1 Python 🦆

2 Thonny

3

1	2
3	4

4 Proměnné

5 Funkce

6 🐢



Funkce – matematika

V matematice jsou funkce zobrazeními. Obvykle pracujeme s čísly, vektory a maticemi, ale není problém zobecnit na jakékoliv množiny.

Funkce – programování

V programování je funkce pojmenovaný kus kódu, na který se lze jinde odkázat a spustit ho. Často má argumenty (vstupy) a vrací nějakou hodnotu (výstup), podobně, jako v matematice.

Matematicky

Definujeme funkci : $\mathbb{R}^2 \mapsto \mathbb{R}$ předpisem  $(x, y) = \frac{(x+y)^2}{3}$.

Matematically

Definujeme funkci : $\mathbb{R}^2 \mapsto \mathbb{R}$ předpisem  $(x, y) = \frac{(x+y)^2}{3}$.

V pythonu

```
1 # function heart computes values of function defined above
2 def heart(x,y):
3     z = (x + y) ** 2 / 3
4     return z
5
6 result = heart(3, -2) # output of heart is stored to result
7 result = heart(result, 0) # it can be used again
8 print(result) # and finally we can see the value
```

Matematically

Definujeme funkci : $\mathbb{R}^2 \mapsto \mathbb{R}$ předpisem  $(x, y) = \frac{(x+y)^2}{3}$.

V pythonu

```
1 # function heart computes values of function defined above
2 def heart(x,y):
3     z = (x + y) ** 2 / 3
4     return z
5
6 result = heart(3, -2) # output of heart is stored to result
7 result = heart(result, 0) # it can be used again
8 print(result) # and finally we can see the value
```

Je rozdíl mezi print a result!

Funkce nemusí mít vstup

```
1 import time
2
3 now = time.ctime()
4 print(now)
```

```
1 def support():
2     return "You're doing great!"
```

Vstupy a výstupy

Funkce nemusí mít vstup

```
1 import time
2
3 now = time.ctime()
4 print(now)
```

```
1 def support():
2     return "You're doing great!"
```

Funkce nemusí mít ani výstup...

```
1 import time
2
3 def whatTimeIsIt():
4     print time.ctime()
```


Vstupy a výstupy

Funkce nemusí mít vstup

```
1 import time
2
3 now = time.ctime()
4 print(now)
```

```
1 def support():
2     return "You're doing great!"
```

Funkce nemusí mít ani výstup...

```
1 import time
2
3 def whatTimeIsIt():
4     print time.ctime()
```

...ale formálně stejně vrátí aspoň None.

Funkce s více vstupy

Funkce s více vstupy

```
1 def student(name, lastname, UCO):  
2     print("Student's name is", name, lastname, "and UCO", UCO)  
3  
4 student("Jan", "Bohm", 408849)  
5 student(UCO = 408849, name = "Jan", lastname = "Bohm")
```

Funkce s více vstupy

Funkce s více vstupy

```
1 def student(name, lastname, UCO):  
2     print("Student's name is", name, lastname, "and UCO", UCO)  
3  
4 student("Jan", "Bohm", 408849)  
5 student(UCO = 408849, name = "Jan", lastname = "Bohm")
```

Funkce s výchozími hodnotami

```
1 def me(name = "Jan", lastname = "Bohm", UCO = 408849):  
2     print("Student's name is", name, lastname, "and UCO", UCO)  
3  
4 me("Aneta", "Novakova", 444444)  
5 me()  
6 me(UCO = "000000", name = "Tom")
```

Časté chyby při definování funkcí

- Chybějící dvojtečka
- Špatné odsazení těla funkce (4 mezery)
- Chybějící prázdný řádek za tělem funkce
- Záměna `print` a `return`
- Chybějící nějaký vstupní parametr

Časté chyby při definování funkcí

- Chybějící dvojtečka
- Špatné odsazení těla funkce (4 mezery)
- Chybějící prázdný řádek za tělem funkce
- Záměna `print` a `return`
- Chybějící nějaký vstupní parametr

Příště si řekneme něco o čitelnosti kódu.

Co nás dnes čeká

1 Python 🦆

2 Thonny

3

1	2
3	4

4 Proměnné

5 Funkce

6 🐢

Než začneme *pořádně programovat*, budeme si hrát s 🐢. Turtle je modul pro výuku základních programovacích dovedností.

```
1 import turtle # module math contains all turtle functions
2
3 Leonardo = turtle.Turtle() # creates turtle named Leonardo
4 Leonardo.forward(100) # Leonardo goes forward 100 px
5 Leonardo.right(90) # Leonardo turns right 90 degrees
6 Leonardo.forward(100) # Leonardo goes forward 100 px
7 Leonardo.back(200) # Leonardo backs 200 px
```

Všechny funkce najdete v dokumentaci.

Nejprve importujeme knihovnu turtle, pak vytvoříme želvu s nějakým jménem a můžeme používat příkazy níže – jen turtle nahradíme jménem želvy.

```
1 turtle.forward(xPixels)
2 turtle.backward(xPixels)
3 turtle.left(xDegrees)
4 turtle.right(xDegrees)
5
6 turtle.circle(radius, extentDegrees)
7
8 turtle.pendown()
9 turtle.penup()
10
11 turtle.pencolor("red")
12 turtle.fillcolor("blue")
13 turtle.begin_fill()
14 turtle.end_fill()
```





Jak nakreslit domeček jedním tahem?




Jak nakreslit domeček jedním tahem?


Algorithm 2: domeček jedním tahem


Input:  – objekt třídy želva
size – velikost základny v px

 → size ;


 ↖ $\sqrt{2} \cdot \text{size}$;

 → size ;

 ↖ $\frac{\sqrt{2}}{2} \cdot \text{size}$;

 ↙ $\frac{\sqrt{2}}{2} \cdot \text{size}$;

 ↓ size ;

 ↗ $\sqrt{2} \cdot \text{size}$;

 ↓ size ;

Funkce s

```
1 # function house draws a one-line house
2 # ninja - name of the turtle
3 # size - size of the base of the house
4
5 def house(ninja, size):
6     ninja.forward(size)
7     ninja.left(135)
8     ninja.forward(2**(1/2) * size)
9     ninja.right(135)
10    ninja.forward(size)
11    ninja.left(135)
12    ninja.forward(2**(1/2) * size / 2)
13    ninja.left(90)
14    ninja.forward(2**(1/2) * size / 2)
15    ninja.left(45)
16    ninja.forward(size)
17    ninja.left(135)
18    ninja.forward(2**(1/2) * size)
19    ninja.right(135)
20    ninja.forward(size)
```