

Úvod do jazyka Python

Dominika Regéciová

FIT VUT + Avast

iregociova@fit.vutbr.cz

dominika.regociova@avast.com

Principy programovacích jazyků a OOP (IPP) 2022



Bakalářské práce:

- Sběr dat z honeypotů a jejich využití pro threat intelligence
- Reverzní inženýrství a analýza malwaru
- A další možná témata
- Zadání pro další rok již budou v režii firmy Avast+Norton
- Pro další informace sledujte WIS nebo Discord
- Kontakt: já nebo můj kolega Lukáš Zobal (izobal@fit.vutbr.cz)













Python budete potřebovat pro 2. projekt do tohoto předmětu:

- Verze 3.8 (hlavně pozor na nekompatibilitu s Pythonem 2!)
- Hodnotící testy budou spouštěny na serveru Merlin
- Pokud budete psát projekt někde jinde, doporučuji před odevzdáním na Merlinovi alespoň vyzkoušet

Cílem přednášky je dát vám základ pro práci s jazykem Python a také tipy na další zdroje, kde je možné dohledat si další informace.

Naopak obsahem nebude návod na řešení projektu, ale občas upozorním na nějaké časté chyby a potíže, které jsem zaznamenala při hodnocení projektů z minulých let.

- [Repositář s příklady z dnešní přednášky](#)
- [Záznam přednášky z minulého roku od Petra Zemka](#)
- [Python: oficiální stránky](#)
- [Python 3.8 dokumentace](#)
- [Python: oficiální návod](#)
- [Changelog k verzi 3.8](#)
- [*Real Python: velmi pěkné stránky s mnoha články a návody](#)
- [*PyCoders Weekly: newsletter s tipy na články](#)

Feb 2022	Feb 2021	Change	Programming Language		Ratings	Change
1	3	▲		Python	15.33%	+4.47%
2	1	▼		C	14.08%	-2.26%
3	2	▼		Java	12.13%	+0.84%
4	4			C++	8.01%	+1.13%
5	5			C#	5.37%	+0.93%
6	6			Visual Basic	5.23%	+0.90%
7	7			JavaScript	1.83%	-0.45%
8	8			PHP	1.79%	+0.04%
9	10	▲		Assembly language	1.60%	-0.06%
10	9	▼		SQL	1.55%	-0.18%

Zdroj: <https://www.tiobe.com/tiobe-index/>

Název je reference na britskou komediální skupinu Monty Python

- V roce 1991 jazyk navrhl Guido van Rossum
- Postupem času vznikly tři nekompatibilní major verze, Python (1), Python 2 a Python 3
- Dnes je podporovaná pouze verze 3.7 a vyšší (3.10 vyšla minulý rok)
- **Filosofie Pythonu**



Zdroj: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

- Python je vysokoúrovňový programovací jazyk pro obecné použití
- Je to imperativní jazyk a podporuje různá programovací paradigmatata
- Interpretovaný jazyk (kód je přeložen do bajtkódu podobně jako Java, interpretován virtuálním strojem)
- Vše je objekt
- Dynamický a silně typovaný jazyk
- Mnoho implementací (CPython, PyPy, Jython, IronPython, ...)
- Přenositelný (Windows, Linux, MacOS,...)
- Duck typing
- Místo středníků a složených závorek využívá bílé znaky (tabulátor/mezery)
- Automatická správa paměti (garbage collector)
- Open-source a zadarmo

- NonType: `None`
- Bool: `True`, `False`
- Int: `42`
- Float: `0.14`, `float('inf')`, `float('nan')`
- Complex: `2 + 3j`
- Str: `'Hello FIT!'`
- Bytes: `b'\x68\x65\x6c\x6f'`

- Znaková sada vs kódování
- Jedno-bajtové vs více-bajtové
- Unicode vs UTF-8, UTF-16, UTF-32
- `Str` vs `bytes` v Pythonu

- Seznam (list): `[1, 4.2, 'hola', None]`
- N-tice (tuple): `('Doctor Who', 1963)`
- Množina (set): `{4, 1, 7, 3}`
- Slovník (dict): `{'Rose': 19, 'Martha': 23, 'Dona': 30}`

- Navázání jmen na objekty (objektům přiřazujeme jméno nebo jména)
- Dynamické typování
- Bez explicitní deklarace
- Od verze 3.5 můžeme deklarovat i s typy, přes tzv. hints, které nemění běh programu, pro další typové kontroly, například s mypy

>>> x = 1	# x → 1
>>> x = 'Hola'	# x → 'hola'
>>> x = [1, 2]	# x → [1, 2]
>>> y = x	# x → [1, 2] ← y
>>> x.append(3)	# x → [1, 2, 3] ← y
>>> a	
[1, 2, 3]	
>>> b	
[1, 2, 3]	
>>> y = [4]	# x → [1, 2, 3]; b → [4]

- Aritmetické: `+`, `-`, `*`, `/`, `//`, `%`, `**`, `@`
- Porovnání: `==`, `!=`, `<`, `>`, `<=`, `>=`
- Bitové: `<<`, `>>`, `|`, `&`, `~`, `^`
- Logické: `and`, `or`, `not`
- Přiřazení: `=`, `:=`, `+=`, `-=`, `*=`, `/=`, `//=`, `%=`, `**=`, ...
- Indexování: `[]`
- Slicing: `[:]`
- Volání: `()`
- Ostatní: `in`, `is`

- Přřazení: `x = 42`
- Výraz: `print('My name is', name)`
- If:
`if x > 10:`
 `x = 10`
`elif x < 5:`
 `x = 5`
`else:`
 `print('error')`
- For:
`for pet in ['dog', 'cat', 'panda']:`
 `print('I have', pet, 'as a pet')`
- Break, continue, assert, return, pass

```
def factorial(n):  
    """Returns the factorial of n."""  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n - 1)
```

```
x = factorial(5) # 120
```

- First-class objekt
- Funkce mohou být vnořené
- Defaultní argumenty
- Názvy parametrů
- Proměnný počet argumentů

```
... # A
def spam():
    ... # B
    def eggs():
        ... # C
        print(x)
```

- Lexikální (statický) rozsah platnosti
- LEGB: konzistentní pravidlo pro vyhodnocování rozsahu platnosti
 - 1 Local
 - 2 Enclosing
 - 3 Global
 - 4 Built-in
- If, for, while nevytváří nový rozsah platnosti
- Explicitní deklarace přes klíčová slova `global` a `nonlocal`

- Globální proměnné existují do konce programu
- Lokální do konce funkce
- Explicitní smazání přes `del`

- Příklad struktury balíčku

```
network/  
    __init__.py  
    socket.py  
    http/  
        __init__.py  
        request.py  
        response.py  
        ...  
    bittorrent/  
        __init__.py  
        torrent.py  
        bencoding.py  
        ...  
    ...  
from network.http.request import Request
```

Import a single module.

```
import time
```

Import multiple modules at once.

```
import os, re, sys
```

Import a module under a different name.

```
import multiprocessing as mp
```

Import a single item from a module.

```
from threading import Thread
```

Import multiple items from a module.

```
from collections import namedtuple, defaultdict
```

Import everything from the given module. (Use with caution!)

```
from email import *
```

```
from math import sqrt
```

```
class Point:
```

```
    """Representation of a point in 2D space."""
```

```
    def __init__(self, x, y):
```

```
        self.x = x
```

```
        self.y = y
```

```
    def distance(self, other):
```

```
        return sqrt((other.x - self.x) ** 2 +  
                    (other.y - self.y) ** 2)
```

```
a = Point(1, 2)
```

```
b = Point(3, 4)
```

```
print(a.distance(b)) # 2.828427124746190
```

- Vytváření instancí a jejich inicializace
- Metoda vs funkce
- Třídy jsou first-class objekty
- Vše je veřejné (public)
- Vše může být předefinované (overridden)
- Každá třída automaticky dědí od třídy object
- Vícenásobná dědičnost, method resolution order (MRO)
- Lze volat metody z bazových tříd
- Instanční proměnné vs proměnné třídy
- Metody instance vs metody třídy vs statické metody

- Vytváření instancí (`__new__()`, `__init__()`)
- Uložení dat instance v paměti (`__dict__`, `__slots__`)
- “Interní” (`_`) a pseudo-privátní (`__`) atributy
- Speciální metody (`__$method__()`), předefinování operátorů
- Použití `super` pro vícenásobnou dědičnost
- Finalizér (`__del__()`)
- Změna procesu vyhledávání atributů (`__getattr__()`, `__getattribute__()`)
- Protokoly, duck typing
- Rozhraní, abstraktní bazové třídy
- Třídy mohou být vytvořeny a rozšiřovány za běhu
- Třídy jsou instancemi metatříd

```
# Raising an exception:  
raise IOError('not enough space')  
  
# Exception handling:  
try:  
    # code  
except IOError as ex:  
    # handle a specific exception  
except:  
    # handle all the other exceptions  
else:  
    # no exception was raised  
finally:  
    # cleanup actions, always executed
```

Bad:

```
f = open('file.txt', 'r')
contents = f.read()
f.close()
```

Better:

```
f = open('file.txt', 'r')
try:
    contents = f.read()
finally:
    f.close()
```

The best:

```
with open('file.txt', 'r') as f:
    contents = f.read()
```

- Jazykové idiomy
- *Pythonic* vs *unpythonic* - používání idiomů v Pythonu
- Zen Pythonu (`import this`)

```
# Unpythonic
```

```
i = 0
```

```
while i < len(items):
```

```
    print(items[i])
```

```
    i += 1
```

```
# Pythonic
```

```
for item in items:
```

```
    print(item)
```


- Formátování řetězců (f-strings, Python 3.6)

```
name = 'Joe'  
item = 'bike'  
print(f'Hey {name}, where is my {item}?')
```

- Anonymní (lambda) funkce

```
people.sort(key=lambda person: person.name)
```

- Vytvoření seznamu, množiny, slovníku pomocí comprehensions (porozumění)

```
list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
squares = [x ** 2 for x in list if x % 2 == 0]  
# [4, 16, 36, 64, 100]
```

- Podmíněný výraz

```
cost = cheap if price <= 100 else expensive
```

- Zřetěžené porovnávání

```
if 1 < x < 5:  
    # ...
```

- Oddělovače číslic (Python 3.6)

```
1_483_349_803
```

- Rozbalení kolekcí

```
head, *middle, tail = [1, 2, 3, 4, 5]
```

- Mroží operátor (Python 3.8)

```
# Loop over fixed length blocks  
while (block := f.read(256)) != "":  
    process(block)
```

- Generátory

```
def fibonacci():  
    a, b = 0, 1  
    while True:  
        yield a  
        a, b = b, a + b
```

```
for fib in fibonacci():  
    print(fib)  
    if fib > 100:  
        break
```

- For cyklus s else větví

```
for item in collection:  
    if item == 5:  
        break  
else: # ?!  
    print("not found")
```

- Modifikovatelné defaultní argumenty

```
def foo(x=[]):  
    x.append(4)  
    return x  
print(foo([1, 2, 3])) # [1, 2, 3, 4]  
print(foo()) # [4]  
print(foo()) # [4, 4] ?!
```

- Unicode identifikátory

π = 3.1415

- Zpracování textu (re, json, xml, csv, base64)
- Datové typy (datetime, collections, dataclasses)
- Konkurentní programování (threading, multiprocessing, asyncio)
- Operační systém a souborový systém (os, shutil, tempfile)
- IPC a síťová komunikace (signal, mmap, selectors, socket)
- Internetové protokoly (urllib, email, smtplib, ipaddress)
- Komprese dat (zipfile, tarfile, gzip)
- Kryptografie (hashlib, hmac, secrets)
- Obdoba funkcionálního programování (itertools, functools)
- Vývoj (unittest, doctest, venv)
- Debuggování a profilování (pdb, timeit, dis)
- Ostatní (logging, argparse, ctypes)

- Python Package Index
- <https://pypi.org/>
- `$ pip install <package_name>`
- Oficiální repositář balíčků pro Python
- Přes 200 000 balíčků
- Je možné vytvořit a zveřejnit vlastní balíček
- Lze vytvořit i vlastní privátní repositář

- Čistá a jednoduchá syntax
- Jednoduchý na naučení
- Vysoká úroveň abstrakce, vyšší produktivita, vhodné na prototypování
- Užitečné vestavěné datové typy
- Elegantní a flexibilní modulový systém
- Velmi dobrá standardní knihovna (+ PyPI)
- Podpora vícero paradigmat
- Generické programování (duck typing)
- Široce používán

- Není vždy dostatečně rychlý pro výpočetně náročné úlohy
- Rovněž není vhodný pro úlohy s vysokou paměťovou náročností
- Omezený paralelismus s vlákny (GIL: Global Interpreter Lock)
- Nejsou zde konstanty jak je známe z jiných jazyků (klíčové slovo `const`)
- Obecně přenositelný, ale některý kód je specifický pro daný OS
- Nekompatibilita mezi verzemi 2 a 3

- Všechno je veřejné
- Nesystematická dokumentace
- Důležitá role tabulátorů a mezer
- Standardizace
- Podpora tzv. "monkey patching"
- Nevhodný pro nízko úrovněvé programování
- Dynamicky typovaný