

JAK TESTOVAT

rukama, očima...

...

...

...

...

...

...programem!

Jednotkové testy / Unit tests

„Jednotka“ - funkce, třída...

MODULY

Jmenný prostor (namespace)

1. V Pythonu je všechno objekt
2. Každý objekt má svůj jmenný prostor

Funkce `dir` vypíše obsah jmenného prostoru

```
dir()
```

```
a=5
```

```
dir()
```

```
dir(a)
```

```
import math
```

```
dir(math)
```

Jak použít („importovat“) modul

1.způsob:

```
import math  
a = math.sqrt( 2 )
```

```
import math as mm  
a = mm.sqrt( 2 )
```

2.způsob:

```
from math import sqrt  
a = sqrt( 2 )
```

3.způsob:

```
from math import * # => tohle NEDELEJTE!  
pi  
pi = sqrt( 2 )  
pi  
math.pi
```

Jak vytvořit modul

= jen uložit zdrojový kód, jako jakýkoliv jiný

Příkazy uvnitř modulu

Provádí se jen při prvním příkazu import

Proměnná `__name__`

V hlavním programu má hodnotu `'__main__'`,
jinak jméno modulu.

```
if __name__ == '__main__':
```

Proč vytvářet moduly

- rozdělení programu na části
- oddělení kódu, který bychom mohli použít i v jiných projektech

Světlá strana

The Python Standard Library

<https://docs.python.org/3/library/>

- součást instalace Pythonu pro Windows
- ...v Linuxu se doinstalovává obvyklými prostředky
- všestranná knihovna
- moduly napsané v Pythonu i v jazyku C

Temná strana

...později.

Zajímavé moduly

`datetime` : Basic date and time types

`math` : Mathematical functions

`random` : Generate pseudo-random numbers

`os.path` : Common pathname manipulations

= už jsme viděli, když jsme zjišťovali existenci souboru

`turtle` : Turtle graphics

`urllib.request` : Extensible library for opening URLs

`html.parser` : Simple HTML and XHTML parser

`tkinter` : Python interface to Tcl/Tk

Mnoho dalších (ne-standardních) doinstalovatelných

`matplotlib`

Balíček

Více souborů/modulů, adresářová struktura.

Soubor `__init__.py` v každém (pod)adresáři.

`__init__.py`

- může být prázdný
- příkaz

```
__all__ = ["aaa", "bbb", "ccc"]  
= seznam modulů pro příkaz  
from balik import *
```

V Programování 1 NE.

I když přece jen...

- Balíčky jsou užitečné
- Můžeme využívat toho, co už někdo naprogramoval
- Nemusíme vynalézat kolo
- The Python Package Index (PyPI) <https://pypi.org/>

204,421 projects 1,536,788 releases 2,288,324 files
(k 10.12.2019)

- Můžeme sami přispět a ostatní mohou využívat naše balíčky

I když přece jen...

- Balíčky jsou užitečné
- Můžeme využívat toho, co už někdo naprogramoval
- Nemusíme vynalézat kolo
- The Python Package Index (PyPI) <https://pypi.org/>

204,421 projects	1,536,788 releases	2,288,324 files	(k 10.12.2019)
274,367 projects	2,218,437 releases	3,549,455 files	(k 26.11.2020)

- Můžeme sami přispět a ostatní mohou využívat naše balíčky

Jenomže (Temná strana)

- Jak moc důvěřujete cizímu kódu?
- (Cizí) balíčky mohou vyžadovat další (cizí) balíčky
- Balíčky se vyvíjejí a mohou mít různé verze
(stejně jako všechno ostatní)
- Různé verze balíčků nemusí být zpětně kompatibilní
= konec klidného spaní. viz "Dependency hell"

Zpětná kompatibilita

JAK TESTOVAT 2

modul unittest

- součást Standardní knihovny
- klon JUnit od Kenta Becka a Ericha Gammy
<https://web.archive.org/web/20150315073817/http://www.xprogramming.com/testfram.htm>
- dříve nazývaný PyUnit
<http://pyunit.sourceforge.net/pyunit.html>
- existuje řada dalších podobných modulů

testovací případ

- měl by být nezávislý na okolí (všechno si připravit sám)
- podtřída třídy `unittest.TestCase`
- testy jsou funkce začínající `test...`
- `assert`
- `unittest.main()` spustí všechny testy
- `setUp(self)` a `tearDown(self)`

Test Driven Development (TDD)

- nejdříve napsat testy
- přesvědčit se, že všechny testy selžou
- pak psát kód, dokud všechny testy neprojdou

Positivní testy x Negativní testy

- když má program selhat, měl by selhat co nejdříve a mělo by to být slyšet
- Selhání (Failure) vs. Chyba (Error)
- výjimky vs. chybové kódy

`isinstance(s, str)`

Např.: `assert isinstance("1234", int)`

`self.assertRaises(výjimka, funkce, parametry)`

Např.: `self.assertRaises(ValueError, int, "abc")`

Další druhy testů

jednotkové testy

testy jednotlivých částí

integrační testy

jestli části programu budou spolupracovat

regresní testy

jestli program ještě funguje tak, jak fungoval včera

Funkční a nefunkční požadavky

CI: Continuous integration (Průběžná integrace)

Poznámka

Jak importovat modul z určité cesty

```
import sys
    sys.path.append("D:/Tom") # pozor na lomítka!
        import mujmodul
```