JAK TESTOVAT

Odbočka

Jak importovat modul z určité cesty

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

Jednotkové testy / Unit testy

„Jednotka" – funkce, třída...

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)
```

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

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: Countinuous integration (Průběžná integrace)

JEŠTĚ K FUNKCÍM...

Funkce jsou objekty

takže se dají ukládat do proměnných

```
tisk = print
tisk(f"{2}*{5}={2*5}")
```

...i předávat jako parametry

```
def tabulka(f):
    for i in range(10):
        print(f"{i:2d}:{f(i):+.2f}")
        # :+.2f = znamenko, 2 desetinna mista
import math
tabulka( math.sin )
tabulka( math.cos )
```

Lambda funkce

lambda calculus, Alonzo Church, 1930

```
tabulka( lambda x: 1/(x+1) ) # 1/(x+1), NE 1/x!
tabulka( lambda x: x*x )

pricti1 = lambda x:x+1
pricti1(27)
   28

soucet = lambda x,y: x+y
soucet(12,25)
   37
```

Lambda funkce 2

je to jediný výraz
nemůžou obsahovat příkazy

známy též jako: anonymní funkce nebo lambda-výrazy

použití: funkce vyžadované jako parametr:

```
lidi
sorted( lidi )
sorted( lidi, key = lambda x: x[1]+x[0] )
```

Closure

funkce může vrátit funkci...

...která si zachová přístup k proměnným nadřízené funkce

```
def pricitacka( kolik ):
    def f(x):
        return x+kolik
    return f

p1 = pricitacka(1)
p1(10)
  11
```

k čemu to je?

- vyhnout se používání globálních proměnných
- ukrývání dat

Closure 2

```
def soucet():
    N = 0
    def secti(x,y):
        nonlocal N
        N += 1
        if N>3:
            print("Uz scitam moc dlouho!")
            return -1
        return x+y
    return secti
```