CHYBY

Každý dělá chyby!

(Jenom ne každý si to uvědomuje.)

Chyby v programech

1. syntaktické chyby (syntax error)
   = ty ohlásí překladač
   = před spuštěním programu
2. běhové chyby (runtime error)
   = ty ohlásí program při běhu
   = až k nim program dojde (pokud)
3. sémantické chyby (semantic error)
   = ty neohlásí nikdo
   (až zklamaný uživatel, pokud si jich všimne)

Neodhalené chyby mohou být velmi drahé!

Mars Climate Orbiter (125 mil. USD)

https://mars.nasa.gov/internal_resources/818/
https://solarsystem.nasa.gov/missions/mars-climate-orbiter/in-depth/

*Vyšetřování zjistilo, že software dodaný společností Lockheed Martin počítal celkový impuls dodaný tryskami sondy **v imperiálních jednotkách (lb·s)**, zatímco software pro výpočet trajektorie dodaný NASA očekával tyto data **v jednotkách SI (N·s)**. Vzhledem k převodnímu faktoru mezi lbs a Ns, který je 4,45, byla takto vypočítaná trajektorie chybná. Sonda se tak namísto plánovaného nejnižšího bodu oběžné dráhy 226 km měla dostat do vzdálenosti pouhých 57 km od povrchu planety. Atmosféra (ač řídká) ji spálila už někde mezi 80 až 90 km od povrchu.*

*(**https://cs.wikipedia.org/wiki/Mars_Climate_Orbiter**)*

Ariane flight V88 (370 mil. USD)

https://en.wikipedia.org/wiki/Ariane_flight_V88
https://www-users.cse.umn.edu/~arnold/disasters/ariane5rep.html
https://se.inf.ethz.ch/~meyer/publications/computer/ariane.pdf

The launch ended in failure due to multiple errors in the software design: ***dead code, intended only for Ariane 4****, with inadequate protection against* ***integer overflow*** *led to an exception handled inappropriately, halting the whole otherwise unaffected inertial navigation system. This caused the rocket to veer off its flight path 37 seconds after launch, beginning to disintegrate under high aerodynamic forces, and finally self-destructing via its automated flight termination system. The failure has become known as one of the most infamous and expensive software bugs in history.[2] The failure resulted in a loss of more than US$370 million.*

Syntaktické chyby

```
>>> if a=b=2:

SyntaxError: invalid syntax

>>> else:

SyntaxError: invalid syntax

>>> deff F():

SyntaxError: invalid syntax
```

Kontrola typů

...je v Pythonu až za běhu!

Takže na chybu nepřijde překladač před spuštěním,
ale až program, když na ni dojde.

ALE (dobrá zpráva)...

Typové anotace

Python 3.5 a novější

```
a: int = 5
b: str = "abcd"

   def delka( slovo: str ) -> int:
       return len(slovo)
```

Python sám nekontroluje, ale externí nástroje ano!
(Visual Studio, Visual Studio Code, PyCharm...)

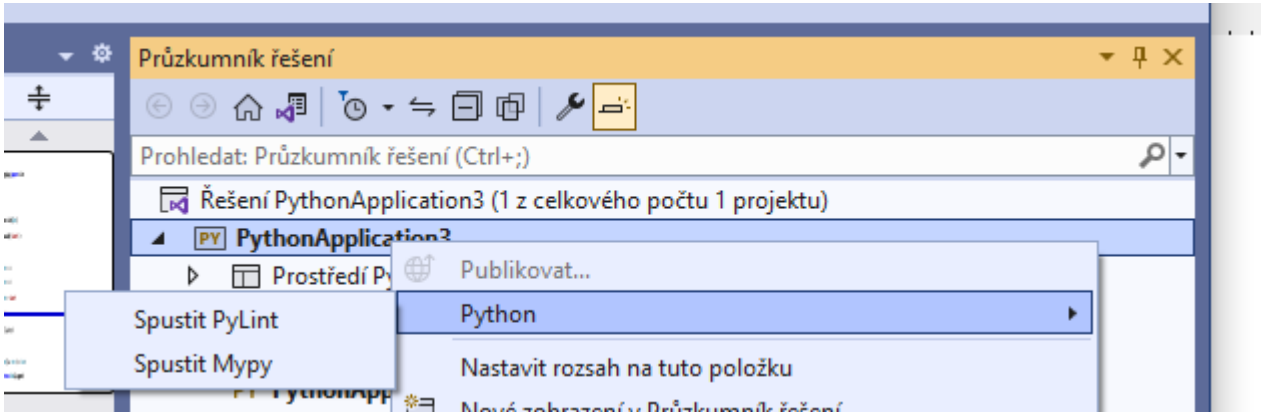
Více: viz PEP483, PEP484, PEP526…
https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html

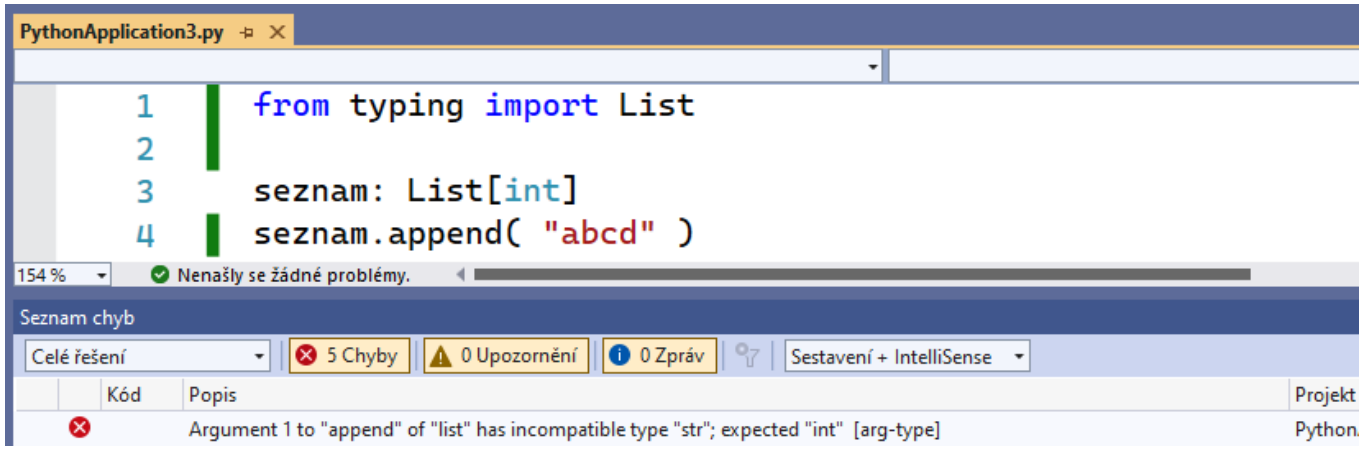
Typové anotace ve VSCode

- (Ctrl+Shift+P) Python: Select Linter mypy
- ...případná instalace mypy
- kontroluje a) automaticky při uložení
  b) (Ctrl+Shift+P) Python: Run Linting

Typové anotace ve Visual Studiu

Průzkumník řešení / Aplikace / RMB / Python / Spustit Mypy





Výjimka

= zpráva o běhové chybě

```
>>> a
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    a
NameError: name 'a' is not defined

>>> a = 5
>>> a[1]
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    a[1]
TypeError: 'int' object is not subscriptable
```

Typ výjimky

Výjimka je objekt
a NameError, TypeError …apod. je jeho typ.

Obsahuje i další informace, jako text zprávy

```
name 'a' is not defined
```

nebo místo, kde k chybě došlo (celý zásobník volání)

```
File "<pyshell#2>", line 1, in <module>
   a[1]
```

.

Jak odchytit výjimku

```
>>> while True:
        try:
            x = int(input("Zadej cislo: "))
            break
        except ValueError:
            print("   Zkus to znovu...")

Zadej číslo:
   Zkus to znovu...
Zadej číslo: sss
   Zkus to znovu...
Zadej číslo: ghgfdhgfd
   Zkus to znovu...
Zadej číslo: 95
>>> x
95
```

Ještě k except

- Lze napsat více bloků except,
  každý pro jiný typ výjimek
- Lze také neuvést žádný typ
  a tím odchytit jakoukoli výjimku, nedělejte to!

(Každý by měl řešit jen ty problémy, kterým rozumí.)

(Pokud se někde dozvíme o chybě,
měli bychom být rádi a nezahazovat tu informaci!)

Ještě k except

Nastalou výjimku si můžeme uložit do proměnné...

```
>>> while True:
        try:
            x = int( input( "Zadej číslo: " ) )
            break
        except ValueError as vyjimka:
            print( "To nebylo číslo..." )
            print( type(vyjimka) )
            print( vyjimka.args )
            print( vyjimka )
            v = vyjimka
            # vyjimka je lokalni, v je globalni
```

Příklad *(podle thinkcspy3)*

```
def existuje(filename):
    try:
        f = open(filename)
        f.close()
        return True
    except FileNotFoundError:
        return False
```

NEBO lépe

```
import os
...
return os.path.isfile("c:/temp/testdata.txt")
```

Výjimkami bychom měli ošetřovat jen **výjimečné** situace.

Finally

Už víme, že soubory je potřeba zavírat...

```
g = open("d:/e.txt","w")
try:
    for i in range(10):
        x = int(input("zadej cislo:"))
        # tady ^ se to může rozbít
        g.write( str(x)+" " )
finally:
    g.close()
```

...tak to můžeme zajistit pomocí finally.

Vyvolání výjimky

proč bychom to měli chtít?

chceme ohlásit výjimečnou situaci,
kterou sami neumíme vyřešit

Například:

funkce má ze vstupu přečíst číslo (a není tam)
máme zjistit údaje z webu a server neodpovídá
...

Vyvolání výjimky

syntaxe:

```
raise ValueError ("Počet trpaslíků < 0")
```

Všechny typy výjimek jsou odvozeny od třídy

BaseException

, můžeme si odvodit vlastní typy.

výjimky vs. návratové kódy

Assert

```
assert 1<2, "Něco není v pořádku"
...
AssertionError: Něco není v pořádku
```

je zkrácený zápis za

```
if not podmínka:
    raise AssertionError( text )
```

POZOR: assert je příkaz, ne funkce! (i v Pythonu3)
=> bez závorek!

Defenzivní programování

Cokoliv se může pokazit, to se pokazí...
...a je dobré se o tom dozvědět včas.

- ověřování vstupu (chyby od ostatních)
- ...i výstupu (naše vlastní chyby)
- testování vstupních a výstupních podmínek a invariantů
- ověřování situací, které nemohou nastat
- nízká tolerance (co neznáme, to odmítnout)
- ...a další

To, že jsi paranoik, ještě neznamená, že po tobě nejdou.
/Woody Allen/

JAK TESTOVAT

rukama, očima…

… … … … …

rukama, očima…

… … … … … …programem!

Jednotkové testy / Unit testy

„Jednotka" – funkce, třída…

- JAK testovat
  - (rukama, očima,) programem...
- CO testovat
  - okrajové případy

Příklad

Další nástroje

List Comprehensions

- pohodlný způsob, jak vytvořit seznam
- nezůstává po něm žádná proměnná

[výraz for ...*(libovolný počet dalších for nebo if)*]

```
[x*x for x in range(10)]

[x*x for x in range(10) if x**3%10==1]

[ [x*y for x in range(1,11)] for y in range(1,11) ]
```

Analogicky pro dict nebo set:

```
{x: x**2 for x in (2, 4, 6)}  {2: 4, 4: 16, 6: 36}
{x**2 for x in (2, 4, 6)}     {16, 4, 36}
```

Funkce v proměnných

Funkce jako parametry funkcí

Closures

Příklady...