Premenné, funkcie, typy

= nie je “rovná sa!”!

```
a = 1231
```

- reprezentuje **priradenie**. a je *premenná*.

Nie je možné napísať:

```
1231 = a
```

Abstrakcia pomocou premenných

```
draha = 300.0
cas = 30.0

rychlost = draha / cas
```

Premennú môžete použiť namiesto hodnoty s rovnakým výsledkom

```
draha = 300.0
print(draha)
```

Ak to môžeme dať do premennej, hovoríme tomu objekt

Skoro všetko v Pythone je objekt:

```
cele_cislo = 12
text = "Dvanásť"
kvazi_realne_cislo = 1 / 3
zoznam = [1, "dva", 3.0]
```

Priradenie nie je objekt (nemá hodnotu):

```
a = (b = 3)
```

Statement vs expression

- *Statement* (príkaz) reprezentuje niečo čo sa má vykonať
- *Expression* (výraz) reprezentuje hodnotu

```
a = 1
```

Priraď 1 do a $\longrightarrow$ príkaz

```
a = 1 + 1
```

Priraď výsledok výrazu `1 + 1` do a. Výraz je výpočet, ktorého hodnota je 2.

Rôzne typy sa nechovajú rovnako

Funkcia `type` vypíše typ. Napr.:

```
type(12) # int (integer)

a = "Reťazec znakov"
type(a) # str (string)
```

```
cele_cislo = 12
text = "Dvanásť"
kvazi_realne_cislo = 1 / 3
zoznam = [1, "dva", 3.0]
```

Skúsme vyskúšať sčítanie pre tieto veci, napr:

```
print(cele_cislo + kvazi_realne_cislo)
type(cele_cislo + kvazi_realne_cislo)
...
```

Akého typu sú nasledujúce objekty?

```
a = 1.32
b = 1,32
```

Zmena typu je občas možná

```
a = 12 / 5
print(a)

b = int(a)
print(b)

c = str(b)
print(c)
print(type(c))
```

Mimochodom, + je operátor

Operátory v Pythone:

```
+       -       *       **      /       //      %       @
<<      >>      &       |       ^       ~       :=
<       >       <=      >=      ==      !=
```

Rýchla tabuľka významov operátorov

Rekapitulácia

Zatiaľ máme:

- Premenné (`a`, `moja_premenna`)
- Hodnoty (`12`, `"TEXT"`, ...)
- Operátory (`+`, `-`, ...)
- Príkazy (`a = 12`)

Čo sú `print()`, `type()`, `int()`, ... -> veci so zátvorkami?

Viac abstrakcie -> funkcie

```
def vypis_dvakrat(x):
    print(x)
    print(x)


vypis_dvakrat("HA!")
vypis_dvakrat(12)
```

Nezabudnite na `:` a indentovaný blok inak dostanete `SyntaxError`.

```
def vypis_a_scitaj(a, b):
    print(a, "+", b)
    return a + b

vysledok = vypis_a_scitaj(1, 3)
print(vysledok)
```

Ak vynecháte `return`, je to akoby ste napísali `return None`:

```
def vypis_dvakrat(x):
    print(x)
    print(x)

vysledok = vypis_dvakrat(12)

print(vysledok)
print(type(vysledok))
```

Mimochodom, čo je `None`

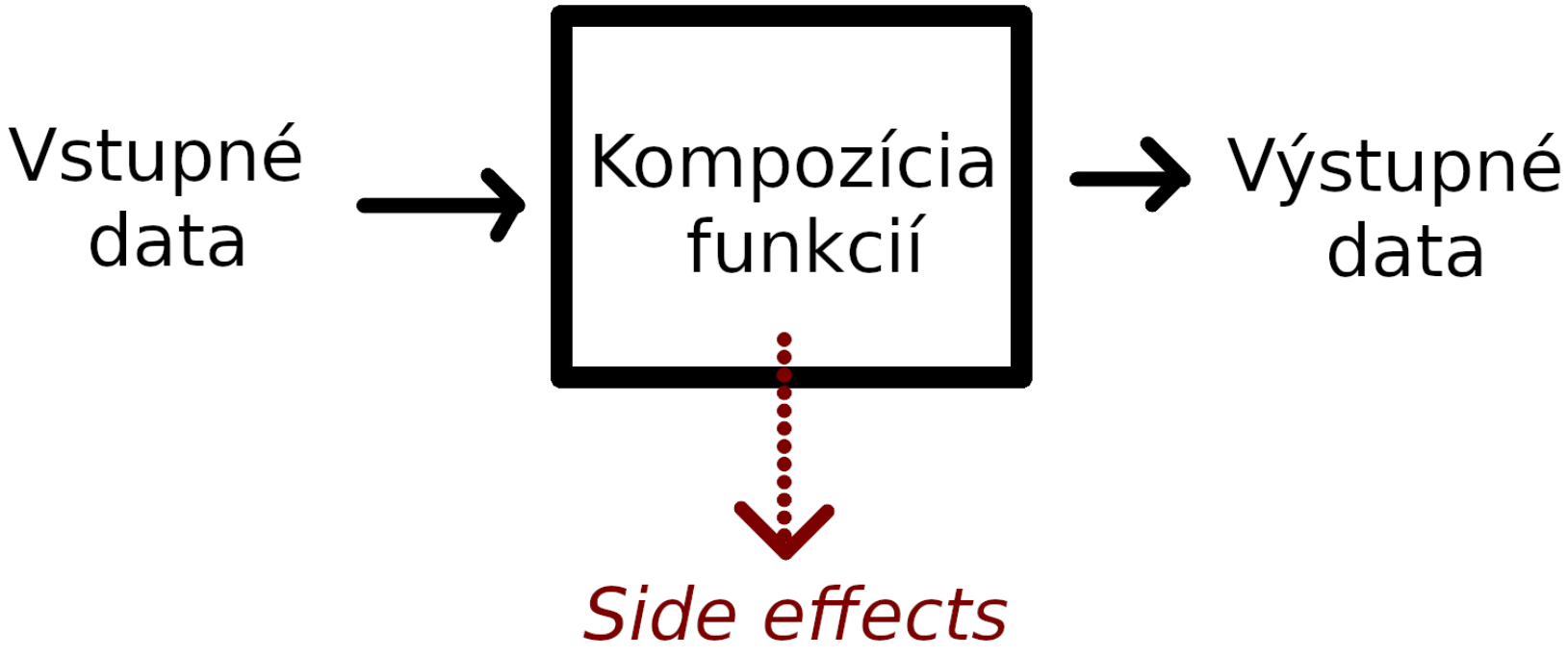
```
print(type(None))
```

`None` je len jeden.

Kompozícia

Funkcie je možné skladať:

```
print(type(vypis_dvakrat(3)))
```



Obr. 1: Program ako kompozícia funkcií

Parametre funkcie sú lokálne premenné

```
def plus_2(a):
    c = a + 2
    return c

a = 1
b = plus_2(3)
print(b)
print(a)
print(c) # chyba
```

Parameter a vo funkcii zakryje globálnu premennú a.

Parametre funkcie sú lokálne premenné, ale...

```
def a_plus_2():
    c = a + 2
    return c

a = 1
b = plus_2()
print(b)
```

Parametre funkcie sú lokálne premenné, ale...

Toto neurobí to čo si možno myslíte:

```
def a_is_a_plus_2(a):
    a = a + 2

a = 1
a_is_a_plus_2(a)
print(a)
```

Keby toto funguje, tak by sme Python nepoužívali:

```
def a_is_a_plus_2():
    a = a + 2

a = 1
a_is_a_plus_2()
print(a)
```

Príkaz, ktorý, dúfam, nikdy potrebovať nebudete

```
def a_is_a_plus_2():
    global a
    a = a + 2


a = 1
a_is_a_plus_2()
print(a)
```

Je funkcia objekt?

```
def vypis_dvakrat(x):
    print(x)
    print(x)

fun = vypis_dvakrat
fun(3)
```

Pozor na zátvorky!!!

```
fun = vypis_dvakrat(3) # cislo
fun = vypis_dvakrat # funkcia
```

Funkcia ako argument funkcie

```
def urob_nieco_s_cislom_tri(fun):
    return fun(3)

urob_nieco_s_cislom_tri(vypis_dvakrat)
urob_nieco_s_cislom_tri(print)

def print_type(x):
    print(type(x))

urob_nieco_s_cislom_tri(print_type)
```

Metódy - funkcie trochu inak

V objektovo-orientovaných jazykoch sú funkcie často priamo napojené na objekty.

```
text = "stop"

print(stop.upper())
```

- `upper` je metóda pre `str`
- `obj.x` (aka bodka/tečka) je syntax, ktorá znamená "*x v obj*"
- Metódu môžete chápať ako funkciu, ktorá dostane ako prvý argument objekt, z ktorého je volaná.

```
def upper(string):
    ...
    return upper
```

Nebudeme si všetko programovať -> import

```
import math

print(math.sin(3.14/2))
```

```
import math as m

print(m.sin(3.14/2))
```

```
from math import sin

print(sin(3.14/2))
```

```
from math import *

print(sin(3.14/2))
print(cos(3.14/2))
```

Viac info v knižke Think Python - kapitoly 1-3

Import antigravity

```
import antigravity
```