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)
```

Existujú rôzne typy a nechovajú sa 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]
```

Operátory v Pythone:

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

Rýchla tabuľka významov operátorov

Rôzne typy sa chovajú rôzne

Skúsme vyskúšať sčítanie pre tieto veci, napr:

```
print(cele_cislo + cele_cislo)
print(cele_cislo + text)
print(cele_cislo + kvazi_realne_cislo)
print(cele_cislo + zoznam)
print(text + text)
print(zoznam + zoznam)
print(text + zoznam)
print([text] + zoznam)
...
```

Zmena typu je občas možná

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

b = int(a)
print(b)

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

Niekedy je implicitná

```
d=10
print(type(d))
d=d+0.1
print(type(d))

...
```

Cvičenie 1

- Zistite akého typu sú nasledujúce objekty?

```
a = 1.32
b = 1,32
c = [1, 32]
d = (1, 32)
e = "132"
```

Cvičenie 2

Overte ako sa chovajú operátory `+`, `-`, `*`, `/` pre dvojice týchto typov:

- `str` a `str`
- `list` a `list`
- `int` a `list`
- `int` a `str`
- `str` a `int`

Nápoveda

- `str` je text (string), napr. ‘“Ahoj!”
- `list` je zoznam, napr. `[1, 2, 3]`
- `int` je celé číslo (integer), napr `12`

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`.

To čo do funkcie dávame (`x` v definícii) voláme argument (parameter) funkcie.

Funkcia s viacerými argumentami a návratovou hodnotou

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

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

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

Cvičenie 3

Napíšte funkciu, ktorá:

- vypíše argument a jeho typ
- pričíta jedničku k argumentu a vráti túto hodnotu

Funkcia bez návratovej hodnoty

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))
```

Kompozícia

Funkcie je možné skladať:

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

Mimochodom, čo je `None`

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

`None` je len jeden.

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 = a_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
    print(a)

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)
```

Je funkcia objekt? (Môžeme ju priradiť do premennej?)

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

fun = plus_2
fun(3)
```

Pozor na zátvorky!!!

```
fun = plus_2(3) # cislo
fun = plus_2 # 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)
```

Cvičenie 4

- Vytvorte funkciu, ktorá vezme 1 argument – inú funkciu a zavolá ju dvakrát.
- Vašu funkciu otestujte.

Metódy - funkcie trochu inak

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

```
text = "stop"

print(text.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_string
```

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
```

Mini kvíz

Cvičenie 5

Napíšte funkciu, ktorá premení uhol v stupňoch na radiany a funkciu inverznú. Číslo π nájdete v knižnici `math`:

```
import math
print(math.pi)
```

Turtle!

Knižnica turtle je štandardná knižnica Pythonu (je nainštalovaná s Pythonom) pre deti, ktorá vám dovoľuje kresliť geometrické útvary. My ju použijeme na precvičenie použitia funkcií a premenných. Myšlienka turtle je, že príkazmi (konkrétne volaniami metód) ovládate pohyb korytnačky (želvy), ktorá pohybom po "papieri" kreslí čiary. Korytnačka môže ísť dopredu, dozadu, otáčať sa, prípadne presunúť sa na konkrétne miesto na papieri.

Dokumentácia turtle je vynikajúca a pravdepodobne tam nájdete všetko čo chcete vedieť.

Turtle príklad

Tu je príklad (náhodná, nezmyselná kresba):

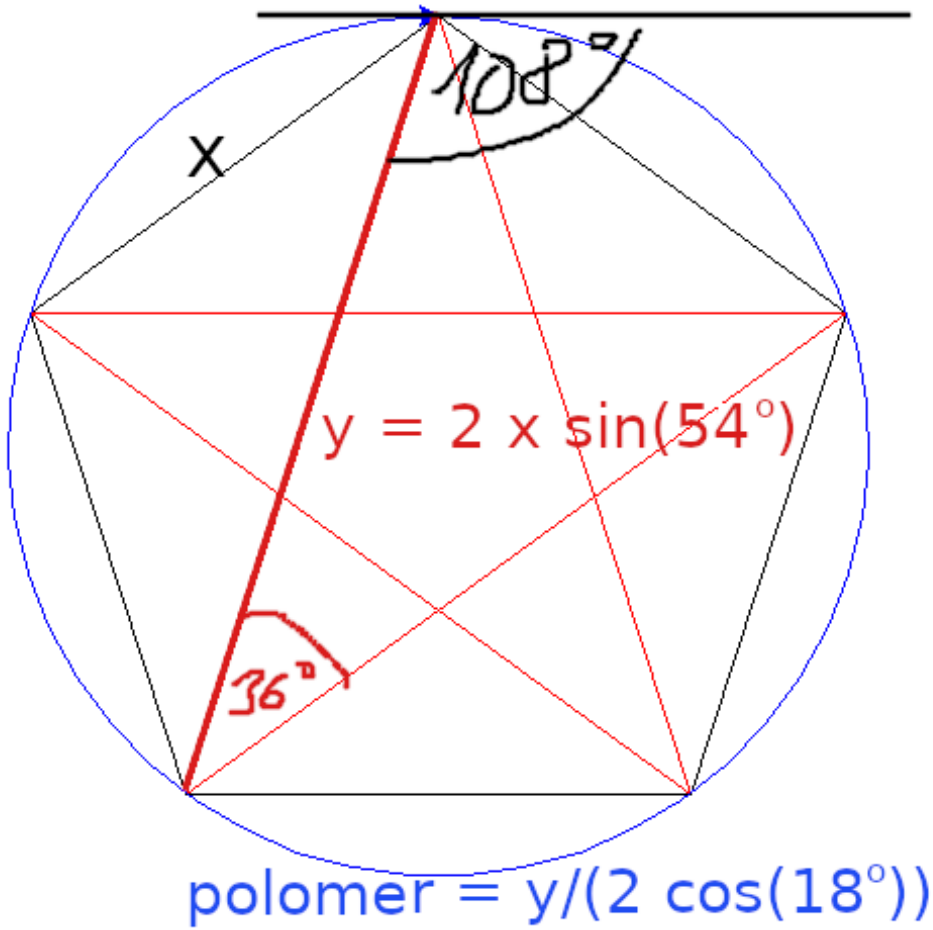
```
import turtle
moje_zelva = turtle.Turtle()

moje_zelva.color("blue")
moje_zelva.shape("turtle")
moje_zelva.speed(1)
moje_zelva.forward(100)
moje_zelva.left(90)
moje_zelva.forward(50)
moje_zelva.circle(60, 100)

moje_zelva.penup()
moje_zelva.right(30)
moje_zelva.backward(150)
moje_zelva.pendown()
moje_zelva.forward(150)
moje_zelva.reset()
```

Cvičenie 6

1. Nakreslite číslo “2”.
2. Nakreslite základné tvary - rovnostranný trojuholník, štvorec, šesťuholník, päťuholník a pentagram.
3. Presuňte váš kód na kresbu geometrických tvarov do funkcií a parametrizujte (uplatnite ako argument vo funkcii) na dĺžku strany.

Cvičenie 7

1. Vyrobte nasledjúci obrázok.



Obr. 1: Relatívne škálovanie dĺžky strany pentagramu a polomeru kružnice.

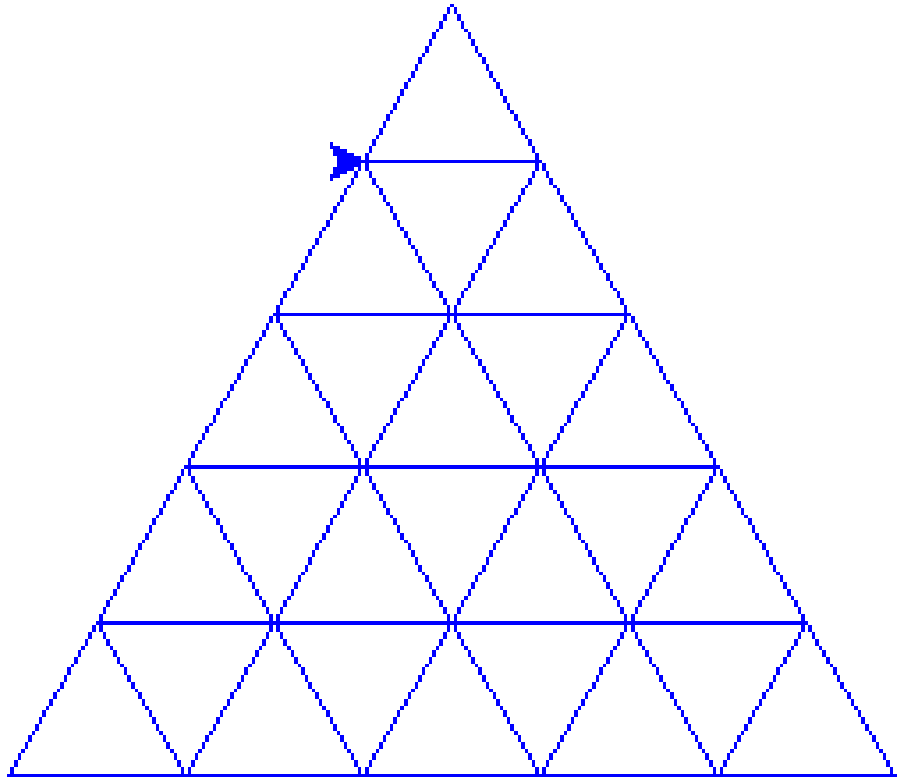
Cvičenie 8

Ukazovali sme si, že funkcie je možné predávať ako argumenty iným funkciam. Ak nechcete písať priveľa rovnakého kódu (známka zlého programátora!), tak si vytvorte pomocné opakovacie funkcie:

```
def run_2(fun):
    fun()
    fun()

def run_3(fun):
    run_2(fun)
    fun()

# ... atď.
```

Cvičenie 8

Vytvorte pyramídu ako na obrázku.



Obr. 2: Pyramída z trojuholníkov

Cvičenie 9

Vyriešte príklady 3.1 a 3.3 z knižky Think Python.