**Třídy a objekty**

V Pythoně je každá hodnota určitým typem (zjišťuje se pomocí funkce ‚typ()‘) a všechny typy jsou objektové – to znamená, že popisují objekty.

A objektovým typům říkáme třída (class).

To znamená, že všechny objekty jsou nějakého objektového typu, neboli třídy a říká se jim **instance třídy.**

**Definice vlastní třídy:**

**class** **Student**:

**pass**

Svojí strukturou je definice třídy podobná definici funkce.

Pomocí konstrukce class Student: jsme vytvořili prázdnou třídu – nový typ ‚Student‘.

**Instance třídy:**

**>>>** fero = Student() *# inštancia triedy*

**>>>** type(fero)

<class '\_\_main\_\_.Student'>

Instance třídy se provádí skrze objektovou proměnou, které odkazuje na jméno třídy se závorkami.

Platí dohody (mezi pythonisty), že všechny nové typy se zapisují při definici s prvním velkým písmenem (‚Student‘). Standartní třídy se uvádějí s malím písmenem (int, float, bool, str, list, tuple).

**Atributy:**

Objekty jsou kontejnery na daty. Pomocí přiřazení můžeme vytvářet v objektu nové soukromé proměnné – **atributy.** Ty se pak chovají úplně stejně jako běžné proměnné, jen se nenacházejí v hlavní paměti – globálním jmeným prostorem – ale v ‚paměti objektu‘.

Atribut objektu vytvoříme tak, že za jméno objektu (instance třídy) ‚fero‘, napíšeme tečku a jméno této soukromé proměnné a vytvoříme jej přidělením:

**>>>** fero.meno = 'Frantisek'

Tímto zápisem jsme vytvořili novou proměnnou – atribut, a přidali jsme ji hodnotu řetězce ‚Frantisek‘

Pokud budeme chtít pracovat s hodnotou tohoto atributu (této proměnné), musíme stejně jak při zadávání uvést nejprve instanci třídy (odkaz na kontejner v kterém se hodnota atributu nachází) za ní tečku a za ní jméno proměnné, pod kterým byla hodnota uložena:

**>>>** fero.meno

'Frantisek'

Atributy jsou mutable, to znamená, že je můžeme kdykoliv měnit.

**Funkce:**

Jedna třída může obsahovat ‚nekonečně‘ instancí třídy (objektů – ‚fero‘, ‚zuzka‘) a každá instance třídy může mít nekonečně atributů (proměnných). Dále každá třída může obsahovat vnitřní funkce (atributy) které pracují s daty uvnitř třídy, nebo externími.

Každý modul je názvem třídy a každá funkce modulu je její atributem.

Funkce rozdělujeme na dva hlavní druhy:

1. Pravá funkce – funkce, které nemění vstupní data, a jen z nich modifikují novou hodnotu
2. Modifikátory – funkce, které něco mění – nejčastěji atribut nějakého objektu.

Příklad pravé funkce:

**def** vypis(st):

print('volam sa', st.meno, st.priezvisko)

**def** urob(m, p):

novy = Student()

novy.meno = m

novy.priezvisko = p

**return** novy

**def** kopia(iny):

novy = Student()

novy.meno = iny.meno

novy.priezvisko = iny.priezvisko

**return** novy

Ani jedna z těchto funkcí nemněmí vstupní parametry.

Příklad modifikátoru:

**def** nastav\_hoby(st, text):

st.hoby = text

print(st.meno, st. priezvisko, 'ma hoby', st.hoby)

Tato funkce mění vstupní parametr ‚st‘ (Student) rozšířením o další atributy.

V objektovém programování platí:

**Objekt** – je kontejner údajů, které jsou soukromými proměnnými objektu (voláme je **atributy**)

**Třída** – je kontejner funkcí, které umí pracovat s objektami (voláme je **metodami**)

**Metody**:

Funkce vytvářené uvnitř třídy jsou umístěné v lokálním prostoru a nazýváme je metoda.

Příklad zapsání výše uvedených globálních funkcí ‚výpis‘ a ‚nastav hoby‘ jako metody – funkce zapsané uvnitř třídy:

**class** **Student**:

**def** vypis(self):

print('volam sa', self.meno, self. priezvisko)

**def** nastav\_hoby(self, text):

self.hoby = text

print(self.meno, self. priezvisko, 'ma hoby', self.hoby)

První parametr metody musí být proměnná v které funkce (metoda) dostane instanci třídy (odkaz na kontejner) s kterou bude dál pracovat.

Dohoda je zde vždy uvádět slovo ‚self‘, jenž odkazuje na třídu ‚Student‘ a její atributy.

K volání této funkce můžeme opět využít formátu ***trieda.metoda(instancia, parametre):***

**>>>** Student.nastav\_hoby(fero, 'gitara')

Ferdinand Fyzik ma hoby gitara

Nebo zkrácenou formu ***instancia.metoda(parametre),*** která je běžnější formou zápisu:

**>>>** fero.nastav\_hoby('gitara')

Ferdinand Fyzik ma hoby gitara

**Magické metody:**

Magické (speciální) metody jsou speciální funkce začínající a končící dvěmi podtržítky a mají vlastní speciální pravidla.

**Magická metoda \_\_init\_\_:**

Je metoda (funkce), která slouží k inicializování atributů daného objektu.

Má tvar:

**def** \_\_init\_\_(self, parametre):

...

Metoda může, ale i nemusí, mít další parametry za ‚self‘. Metoda nic nevrací a nejčastěji obsahuje jen přiřazení. Python tuto metodu (pokud existuje) volá v momentě, když vytváří novou instanci (‚fero‘).

Když tedy zapíšeme instancia = Trieda(parametre), python postupně provede tyto kroky:

1. Vytvoří nový objekt typu třída (zatím prázdný kontejner) – vytvoří se pomocná reference na tento nový objekt.
2. Pokud existuje metoda \_\_iniz\_\_, bude zavolaná s příslučnými parametry
3. Do promněnej Instance přiřadí právě vytvořený objekt

**class** **Student**:

**def** \_\_init\_\_(self, meno, priezvisko, hoby=''):

self.meno = meno

self.priezvisko = priezvisko

self.hoby = hoby

**def** vypis(self):

print('volam sa', self.meno, self. priezvisko)

**def** nastav\_hoby(self, text):

self.hoby = text

print(self.meno, self. priezvisko, 'ma hoby', self.hoby)

Díky \_\_init\_\_ už nepotřebujem funky ‚urob‘, protože tato funkce již očekává uvedené parametry ‚meno‘, ‚priezvisko‘ a nebovynně ‚hoby‘.

**Funkce dir():**

Vrací seznam všech atributů třídy, nebo instance:

**>>> class** **Test**: **pass**

**>>>** dir(Test)

['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',

'\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_',

'\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_',

'\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_',

'\_\_subclasshook\_\_', '\_\_weakref\_\_']

V momentě založení třídy, již obsahuje řadu atrbutů magických funkcí, které se vytvážejí automaticky.

Vždy když zadefinujeme nový atribut, nebo metodu, objeví se i v tomto seznamu dir():

**>>>** t = Test()

**>>>** t.x = 100

**>>>** t.y = 200

**>>>** dir(t)

['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',

'\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_',

'\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_',

'\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_',

'\_\_subclasshook\_\_', '\_\_weakref\_\_', 'x', 'y']

Většinou na konci seznamu, za magickými funkcemi.

Příklad s grafikou:

**import** **tkinter**

**class** **Kruh**:

**def** \_\_init\_\_(self, r, x, y, farba='blue'):

self.r = r

self.x = x

self.y = y

self.farba = farba

**def** vypis(self):

print(f'Kruh(*{*self.r*}*, *{*self.x*}*, *{*self.y*}*, *{*self.farba*!r}*)')

**def** kresli(self):

canvas.create\_oval(self.x-self.r, self.y-self.r,

self.x+self.r, self.y+self.r,

fill=self.farba)

canvas = tkinter.Canvas()

canvas.pack()

a = Kruh(70, 200, 100, 'yellow')

b = Kruh(10, 180, 80)

c = Kruh(10, 220, 80)

zoznam = [a, b, c]

**for** k **in** zoznam:

k.kresli()

**for** k **in** zoznam:

k.vypis()

tkinter.mainloop()

Program nakreslí kruhy a vypíše:

Kruh(70, 200, 100, 'yellow')

Kruh(10, 180, 80, 'blue')

Kruh(10, 220, 80, 'blue')

Rozdíl v dir() mezi třídou ‚Kruh‘ a atributem ‚a‘:

**>>>** dir(Kruh)

['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',

'\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_',

'\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_',

'\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_',

'\_\_subclasshook\_\_', '\_\_weakref\_\_', 'kresli', 'vypis']

**>>>** dir(a)

['\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_',

'\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_',

'\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_',

'\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_',

'\_\_subclasshook\_\_', '\_\_weakref\_\_', 'farba', 'kresli', 'vypis', 'r', 'x', 'y']