Skriptovací jazyky

Datové typy

Python – netypový jazyk

=> nemusíme definovat datový typ

=> dán konkrétní hodnotou

- Proměnné formou odkazu na objekt

```
>>> x = "červená"
>>> y = "zelená"
>>> z = x
>>> print (z)
červená
```

Datový typ přiřazen až při zpracování

```
>>> cislo = 10 #inicializuje proměnnou číslo a přiřadíme k ní číselnou hodnotu
>>> print(type(cislo))
<class 'int'>
>>> print(cislo)
10
>>> cislo = "10" #nyní inicializuje stejnou proměnnou, ale obsahem bude řetězec
>>> print(type(cislo))
<class 'str'>
>>> print(cislo)
10
```

Explicitní definice typu proměnné

```
>>> cislo = str() #přiřazení datového typu str
>>> type (cislo)
<class 'str'>
>>> cislo = int() #přiřazení datového typu int
>>> type (cislo)
<class 'int'>
>>> cislo = int(10) #přiřazení datového typu int zároveň s hodnotou
>>> type (cislo)
<class 'int'>
>>> cislo = str("10") #přiřazení datového typu str zároveň s hodnotou
>>> type (cislo)
<class 'str'>
```

Lokální proměnné v rámci bloku

```
>>> a = 10 #globální proměnná a
>>>
>>> def funkce(): #začátek nového bloku
    a = 20 #lokální proměnná a
    return a

>>> print(a)
10
>>> print (funkce())
20
>>>
```

Základní datové typy

Datový typ	Význam	Příklad
int	Celá čísla (kladná i záporná)	-1992, 95332421, 0
float	Reálná čísla (kladná i záporná)	3.14, 2999341.0, 1923.004
str	Řetězce (posloupnosti znaků)	"10", "Ahoj", "Ahoj světe!"
bool	Pravdivostní hodnoty	True, False

- **integer** – nemá pevný počet bytů, omezen pouze velikostí paměti
- **float** - omezen

```
>>> import sys
>>> sys.float_info #úplné informace o datovém typu float
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.
2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsi
lon=2.220446049250313e-16, radix=2, rounds=1)
>>> sys.float_info.max #maximální hodnota datového typu float
1.7976931348623157e+308
>>> sys.float_info.min #minimální hodnota datového typu float
2.2250738585072014e-308
```

Inf = nekonečno (lze používat v matematických operacích jako „číslo“)

- **str** – posloupnost znaků ohraničená uvozovkami – jednoduchými nebo dvojitými

```
>>> retezec = "slovo v uvozovkach: 'slovo'"
>>> print(retezec)
slovo v uvozovkach: 'slovo'
>>>
>>> retezec2 = 'slovo ve dvojitych uvozovkach: "slovo"'
>>> print (retezec2)
slovo ve dvojitych uvozovkach: "slovo"
```

- **str**
  - jednotlivé znaky řetězce – posloupnost
  - přístup pomocí [] s indexováním od 0

```
>>> retezec = "abc"
>>> print (retezec[0]) #zobrazí první znak řetězce
a
>>> print (retezec[1]) #zobrazí druhý znak řetězce
b
>>> print (retezec[2]) #zobrazí třetí znak řetězce
c
>>> print (retezec[3]) #čtvrtý znak už ale není definovaný
Traceback (most recent call last):
  File "<pyshell#134>", line 1, in <module>
    print (retezec[3]) #čtvrtý znak už ale není definovaný
IndexError: string index out of range
```

- **bool**
  - *True* (*pravda*) a *False* (*nepravda*).
  - !!!!! **Důležité je velké písmeno na počátku slova.**
  - Využití podmínky a cykly

```
>>> 3>2 #například porovnávací operátory vrací odpověď typu bool
True
>>> 2>3
False
```

Převod datových typů

- základní datové typy nelze měnit

```
>>> retezec = "Ahoj" #Definujeme řetězec se slovem Ahoj
>>> print (retezec[0]) #První znak řetězce je A
A
>>> retezec[0] = 'B' #Chceme znak A změnit na B - neúspěšně
Traceback (most recent call last):
  File "<pyshell#144>", line 1, in <module>
    retezec[0] = 'B' #Chceme znak A změnit na B - neúspěšně
TypeError: 'str' object does not support item assignment
```

Převod datových typů

- datový_typ(objekt)

```
>>> int ("45") #Převod řetězce str na celé číslo int
45
>>> int ("Slovo") #Ne každý řetězec lze převést
Traceback (most recent call last):
  File "<pyshell#150>", line 1, in <module>
    int ("Slovo") #Ne každý řetězec lze převést
ValueError: invalid literal for int() with base 10: 'Slovo'
>>> str (912) #Převod celého čísla int na řetězec str
'912'
>>> float (13) #Převod celého čísla int na reálné float
13.0
>>> int(45.99992)#Převod reálného čísla float na celé int
45
```

Převod datových typů

- převod na int dělá trunc => lépe nejprve zaokrouhlit

```
>>> int(45.99992) #Bez zaokrouhlení
45
>>> int(round(45.99992)) #Se zaokrouhlením
46
```

Kolekce

- tuple (n-tice)
- list (seznam)
- dict (slovník) – klíč x hodnota (pole/tabulka)

Kolekce

- tuple (n-tice)

```
>>> "Německo", "Francie", "Itálie", "Španělsko" #Definice n-tice
('Německo', 'Francie', 'Itálie', 'Španělsko')
>>> countries = "Německo", "Francie", "Itálie", "Španělsko" #Uložíme n-tici
do proměnné countries
>>> type (countries) #datový typ je skutečně n-tice tuple
<class 'tuple'>
>>> numbers = 4, #jednoprvková n-tice
>>> type (numbers) #datový typ je skutečně n-tice tuple
<class 'tuple'>
>>> numbers.append(7)
Traceback (most recent call last):
  File "<pyshell#183>", line 1, in <module>
    numbers.append(7)
AttributeError: 'tuple' object has no attribute 'append'
>>> prazdnaNtice = () #Definice prázdné n-tice
>>> type (prazdnaNtice) #datový typ je skutečně n-tice tuple
<class 'tuple'>
>>> print (countries[0], countries[1]) #K prvkům přistupujeme pomocí indexu
Německo Francie
```

Kolekce

- list (seznam)

```
>>> mujseznam = [1,4,9,16,36,49] #Uložíme seznam do proměnné mujseznam
>>> type (mujseznam) #datový typ je skutečně seznam list
<class 'list'>
>>> ['alfa', 'beta', 'gama', 'delta', 'echo'] #seznam naplněný řetězci str
['alfa', 'beta', 'gama', 'delta', 'echo']
>>> ['zebra', 49, -879, 'slon', 2000] #seznam naplněný hodnotami různých datových
['zebra', 49, -879, 'slon', 2000]
>>> [] #toto je prázdný seznam
[]
```

– měnitelné na rozdíl od n-tic

```
>>> seznam = ["a", "b", "c", "d"] #vytvoříme seznam
>>> seznam[2] #zobrazíme prvek seznamu na indexu 2
'c'
>>>
>>> seznam[2] = "3" #narozdíl od n-tic můžeme tento prvek upravit
>>> print (seznam) #zobrazíme seznam po úpravě prvku
['a', 'b', '3', 'd']
```

Kolekce

- list (seznam) – operace
  - *seznam.append(x)* – přidá prvek *x* na konec seznamu,
  - *seznam.remove(x)* – odstraní první výskyt prvku, jehož hodnota je ekvivalentní k *x*,
  - *seznam.insert(i, x)* – přidá prvek *x* na pozici s indexem *i* v seznamu,
  - *seznam.pop(i)* – vrátí hodnotu prvku s indexem i a odstraní tento prvek ze seznamu,
  - *seznam.clear()* – odstraní ze seznamu všechny jeho prvky.

Kolekce

- dict (slovník)
  - definujeme výčtem prvků oddělenými čárkou, které jsou ohraničeny složenými závorkami, tedy {}.
  - *Klíč* je pak od hodnoty k němu přiřazené oddělen dvojtečkou.
  - *Klíče* i *hodnoty* mohou být libovolných datových typů.

Kolekce

- dict (slovník)
  - tabulka x sdružování informací

```
>>> barvy = {"modrá":1, "zelená":2, "červená":3} #slovník, který slouží jako
vyhledávací tabulka
>>> barvy["zelená"] #klíč zelená vrací hodnotu 2
2
>>> notebook = {"CPU":"Intel i5", "RAM":"8GB", "DISK":["SSD 128GB", "HDD 1TB"]}
#druhý slovník slouží ke sdružování informací o našem notebooku
>>> notebook["CPU"] #klíč CPU vrací hodnotu Intel i5
'Intel i5'
>>> notebook["DISK"] #hodnotou pro klíč DISK je list obsahující dva prvky
['SSD 128GB', 'HDD 1TB']
```