

# Složitější proměnné

- **OBJEKT** `zelva.forward(100)`
- **STRING** (textový řetězec) `"abcd"`  
neměnitelná
- **TUPLE** (n-tice) `( "NPRG031", 1 )`  
neměnitelná
- **LIST** (seznam) `[ 1, 2, 5, 3, 2 ]`  
seznam s pořadím
- **SET** (množina) `{ 1, 5, 2 }`  
bez pořadí, každá hodnota je nebo není (max jednou)
- **DICTIONARY** (slovník) `{ 1:"a", 5:"xyz", 2:22 }`  
dvojice klíč:hodnota

# Tuple (n-tice)

- neměnitelná

```
>>> t = (10, 20)
```

```
>>> t[0]
```

```
10
```

```
>>> t[1]
```

```
20
```

```
>>>
```

# Dictionary (slovník)

- Prvky jsou dvojice klíč: hodnota
- "asociativní pole"
- Každý klíč nejvýše jednou
- Uložené bez pořadí, vyhledává se podle klíče

```
>>> d = { 'jablko' : 'apple' }
```

```
>>> d[ 'slovo' ] = 'word'
```

```
>>> d
```

```
{ 'jablko' : 'apple', 'slovo' : 'word' }
```

```
>>> d[ 'jablko' ]
```

```
'apple'
```

# Dictionary (slovník)

## Vymazání prvku

```
>>> del d['jablko']  
>>> d  
  
{ 'slovo': 'word' }
```

## Vytvoření prázdného seznamu

```
>>> d = {} # rychlejší  
>>> d = dict() # pomalejší
```

# Dictionary (slovník)

## Dotaz na existenci klíče

```
>>> 'slovo' in d
True
```

## Cyklus

```
>>> for klic in d:
        print( klic, d[klic] )

slovo word
```

# Dictionary (slovník)

## Indexování neexistujícím klíčem

```
>>> d['slovo']
```

```
word
```

```
>>> d['mrkev']
```

```
...
```

```
KeyError: 'mrkev'
```

# Dictionary (slovník) - klíč

Klíč může být jakákoliv neměnitelná hodnota.

```
>>> d[(25,17,'xyz')] = 'trojice'
```

```
>>> d
```

```
{'slovo': 'word', (25, 17, 'xyz'):  
'trojice'}}
```

# Dictionary (slovník) - proč

Dvojice bychom mohli ukládat i do LIST-u,  
proč potřebujeme DICTIONARY?

```
>>> L = [['slovo', 'word'],  
         [(25, 17, 'xyz'),  
         'trojice']]
```

```
>>> L  
[['slovo', 'word'], [(25, 17,  
'xyz'), 'trojice']]
```



# Dictionary (slovník) - proč

PROČ 1: Možná nehledáme PRVEK toho seznamu, ale jenom jeho klíč (a pak se teprve chceme podívat na hodnotu) - takže nemůžeme použít `in` nebo `find()`

PROČ 2: Vyhledávání v LISTu postupně prochází, vyhledávání v DICTIONARY je pomocí hashování a to je rychlejší ( $O(n)$  vs.  $O(1)^*$ ).

**\*) přibližně, nepřesně, podrobnosti později (Algoritmizace)**

= > lepší třída pro seznamSlov

# Textový řetězec (druhý pohled)

Je neměnitelný (**immutable**),  
pokud chceme část změnit, tak vyrobíme nový  
řetězec.

## Obsah + délka

Možnosti (v různých jazycích):

- pamatovat si délku
- ukončovací znak

Python používá ukončovací znak.

Při dosazování se řetězec **nekopíruje!**

<https://www.laurentluce.com/posts/python-string-objects-implementation/>

# Operátor `in` u typu `string`

```
>>> 'a' in "abcd"
```

```
True
```

```
>>> for x in "abcd":  
        print( x )
```

```
a  
b  
c  
d
```

# Funkce typu string

## find(...):

```
>>> "abcd".find( "c" )  
2
```

(pokud neobsahuje, vrátí -1)

## join(...):

```
>>> "-".join( ['a', 'b', 'c'] )  
'a-b-c'
```

# Funkce typu string

## replace(...):

```
>>> "minimalizace".replace( "min", "max" )  
'maximalizace'
```

## format(...):

```
>>> "a={0} b={1} ab={0}{1}".format( 12, 45 )  
'a=12 b=45 ab=1245'
```

mnoho možností, variant, parametrů...

# Formátovaný řetězec

v Pythonu od verze 3.6

```
>>> a = "xyz"  
>>> b = f"{a} {a} {a}"  
>>> b  
'xyz xyz xyz'
```

Výstup pomocí šablon...

=> **třída Tiskárna**

# Textový soubor

Představa o textovém souboru (nejen v Pythonu):

je to posloupnost znaků

na konci každého řádku je zvláštní znak

ODDĚLOVAČ ŘÁDEK

na konci každého textového souboru je zvláštní znak

UKONČOVACÍ ZNAK SOUBORU

tyto zvláštní znaky **NEPATŘÍ** do množiny hodnot typu

CHAR

jejich smysl je dávat souboru **STRUKTURU**



**Základní akce = vstup/výstup jednoho znaku\*)**

**načte-a-vrátí/zapíše jeden znak\*) a posune se  
v souboru o jeden znak dále. Zpátky NELZE.**

**Po přečtení posledního znaku nelze číst dál.**

# Skutečnost (temná)

řádky textových souborů jsou odděleny dvojicí znaků

CR a LF (`chr(13)` a `chr(10)`)

(! nebo jenom CR nebo jenom LF !

= zdroj SPOUSTY problémů!)

textový soubor je ukončen znakem EOF (`chr(26)`)

(! nebo taky ne, nemusí !)

# Čtení z... a psaní do... souborů

## otevřít

```
>>> f = open( "vstup.txt", "r" )
>>> g = open( "vystup.txt", "w" )

>>> g2 = open( "vystup.txt", "a" )
>>> f2 = open( "vystup.txt", "r+" )
```

## Čtení z... a psaní do... souborů 2

číst:

```
>>> vsechno = f.read()
      nebo
>>> znak = f.read(1) # "" pokud je konec
      nebo
>>> radek = f.readline()
      # "" pokud je konec
      # jinak obsahuje I "\n" (znak 10)
      nebo
>>> for radek in f:
      print( radek )
```

...a další možnosti

## Čtení z... a psaní do... souborů 3

psát:

```
>>> f.write("Jedna radka\n")
```

zavřít:

```
>>> f.close()
```

Proč je důležité zavírat soubory...

=> třída Čtečka (ze souboru)

=> třída Tiskárna (do souboru)