

## ...ještě k objektům:

```
class Static:
    @staticmethod
    def Sedm():
        return 7
    #@staticmethod
    def Osm():
        return 8

print( Static.Sedm() )
print( Static().Sedm() )
print( Static.Osm() )
print( Static().Osm() )
```

```
7
7
8
...print( Static().Osm() )
TypeError: Static.Osm()
takes 0 positional
arguments but 1 was
given
```

...ještě k objektům:

Použití objektů nám dovoluje

nepřepisovat existující zdrojový kód

ale místo toho

odvodit novou třídu,

kde změníme to, co potřebujeme změnit.

Příklad s želvou...



# 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)

soubor neobsahuje znaky ale bajty,  
znak je dlouhý jeden nebo více bajtů

řádky textových souborů jsou odděleny  
dvojicí znaků/bajtů CR a LF (`chr(13)` a `chr(10)`)

(! nebo jenom CR nebo jenom LF !

= zdroj SPOUSTY problémů!)

textový soubor je ukončen znakem/bajtem 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 - kódování

Soubor je posloupnost bajtů, aby z ní vznikly znaky, potřebujeme vědět, jaké **kódování** se má použít.

Výchozí kódování se bere z nastavení operačního systému, můžeme si říci, jaké kódování chceme použít:

```
f = open( "a.txt", "r", encoding="utf8")  
g = open( "b.txt", "w", encoding="utf8")
```



# 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...

# Čtení z... a psaní do... souborů 4

context manager (později):

```
with open( "a.txt", "r" ) as f:  
    for radek in f:  
        print( radek )
```

...na konci bloku with soubor zavře



# 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
```

```
>>> 'bc' 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)

index(...):

```
"abcd".index( "c" )
```

2

(pokud neobsahuje, vyvolá chybu)

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} {5+8}"  
>>> b  
'xyz xyz 13'
```

## Výstup pomocí šablon...

```
>>> sablona = "a={0} b={1}"  
>>> sablona.format( 10,20 )  
'a=10 b=20'
```

(Šablonu můžeme třeba načítat z konfigurace.)

# Příklady

další třídy pro Čtečku a pro Tiskárnu