Příklad funkce

Potřebujeme postupně číst čísla,
která ale mohou být zapsaná tak, že je

- více čísel na jedné řádce
- vstup na více řádek

Takže pro čtení jednoho čísla musíme přečíst celou řádku
a zároveň nesmíme ztratit ta čísla, která tam ještě budou.

```
def PrectiJednoCisloZeVstupu():
```

Dynamické proměnné

- Proměnné, které vznikají (a zanikají) během výpočtu
- Mohou mít různou velikost
- Zvláštní oblast paměti = halda
- Nepotřebné proměnné se automaticky uklízejí
  (ne ve všech jazykách, ale v Pythonu ANO)
  = garbage collector
- Proměnná, která vypadá, že obsahuje třeba seznam,
  ve skutečnosti obsahuje pouze odkaz (adresu do haldy)
- Příklad: list, string, (dlouhé) celé číslo, slovník, objekt
- …ale také spojový seznam (vytvořený z objektů) nebo strom
  (taky vytvořený z objektů)
- prázdný ukazatel None (domluvená hodnota nikam neukazuju)

Objekt (podrobnější pohled)

Obsahuje DATA a FUNKCE

Přístup k nim pomocí tečky

```
>>> a = 5
>>> a.bit_length()
3
>>> zelva.left(90)
>>> zelva.DEFAULT_ANGLEOFFSET
0
```

Skoro všechno v Pythonu je objekt.

Proč objekty (nejen v Pythonu)

Objektová dekompozice

- Rozdělení programu na části, organizace kódu
- Další *CO* to dělá **x** *JAK* to dělá
- Řeší problémy, na které jsme zatím potřebovali globální proměnné (číst vstup, seznam slov…)
- Lze chápat i tak, že obsahují dodatečné parametry svých funkcí…

Objekt obsahuje DATA a FUNKCE

…a ne vše, co obsahuje, je veřejné!

Příklad použití objektů

Napište program,

který čte za vstupu slova a tiskne je do řádek,

které nebudou delší než 30 znaků.

Objekty a třídy (nejen v Pythonu)

Pojmy:

Třída (class) je datový typ (formička)

Objekt je instance (bábovička)

TH je objekt třídy Člověk

Definice třídy

```
class Komplex:
    pass  # dál už nic není
```

Vytvoření objektu

```
k = Komplex() # nový objekt třídy Komplex
k.Re = 3.00
k.Im = 4.00
print( k.Re, k.Im )

3.0 4.0
```

Přidat funkci

- funkce má povinný první parametr `self` = TENHLE objekt, který (ten parametr) se při volání neuvádí

```
class Komplex:
    def abs( self ):
        return (self.Re**2+self.Im**2 )**(1/2)

k = Komplex()
k.Re = 3.00
k.Im = 4.00
print( k.Re, k.Im, k.abs() ) # bez parametru!
```

```
3.0 4.0 5.0
```

Konstruktor

- funkce pro inicializaci nového objektu
- volá se při vytváření objektu
- vždycky se jmenuje `__init__`
- …takže může být jenom jeden (na rozdíl od jiných jazyků)
- má také povinný první parametr `self`
- když konstruktor nenadefinujeme,
  použije se standardní (prázdný) konstruktor

Konstruktor

```
class Komplex:
    def abs( self ):
        return (self.Re**2+self.Im**2 )**(1/2)
    def __init__( self, Re, Im ):
        self.Re = Re
        self.Im = Im

k = Komplex( 3.00, 4.00 )
print( k.Re, k.Im, k.abs() )
```

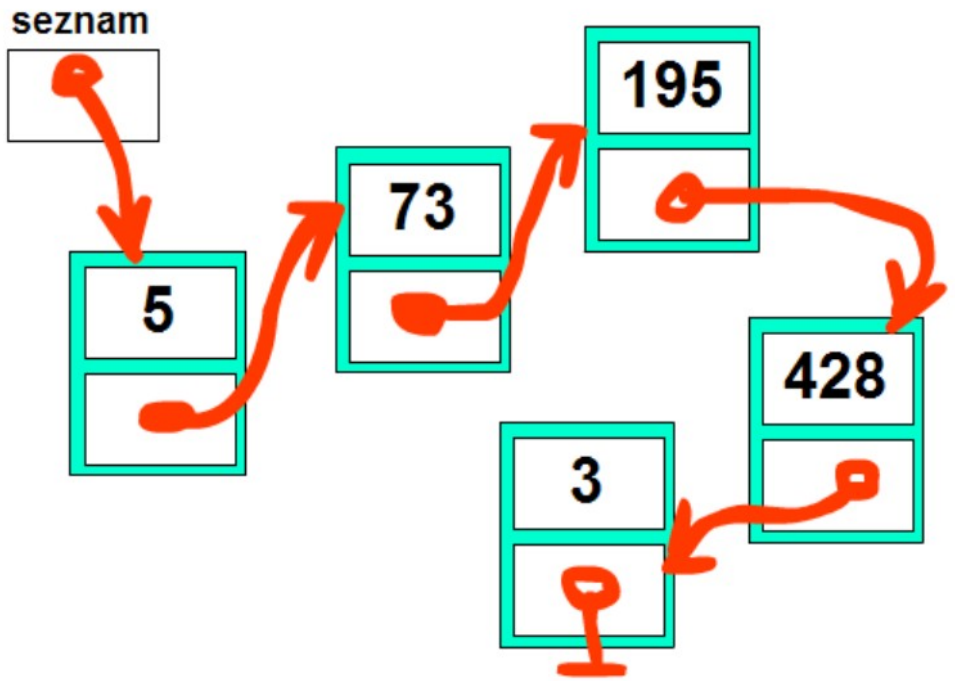
```
3.0 4.0 5.0
```

SKOK

(k objektům se ještě vrátíme)

Lineární spojový seznam

- složený z prvků
- každý prvek obsahuje hodnotu a odkaz na další prvek



Lineární spojový seznam

```
class Prvek:
    def __init__(self, x, dalsi):
        self.x = x
        self.dalsi = dalsi

seznam = Prvek(1,
               Prvek(2,
                     Prvek(3, None )))
def Vytiskni(s):
    while s!= None:
        print( s.x, end=' ' )
        s = s.dalsi

Vytiskni( seznam )
```

```
1 2 3
```