Funkce

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

Seznamy

Vytvoření seznamu

- Syntaxe:

```
[první_položka_seznamu, druhá_položka_seznamu, ..., poslední_položka_seznamu]
```

- Položky seznamu jsou reference na (libovolné) objekty, které reprezentují tyto položky.
- Seznam je mutabilní.
- Příklady:

  L = [1, 2, 3, 4]

  L = ["s", "p", "a", "m"]

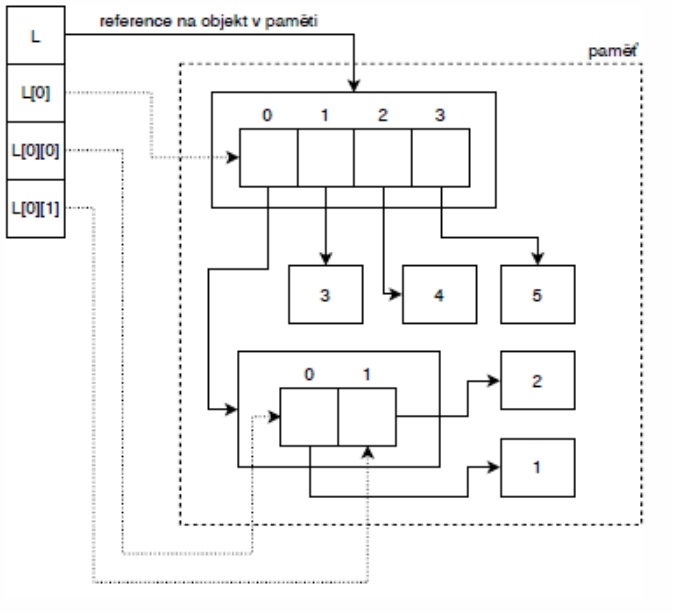
  L = ["spam", "ham"]

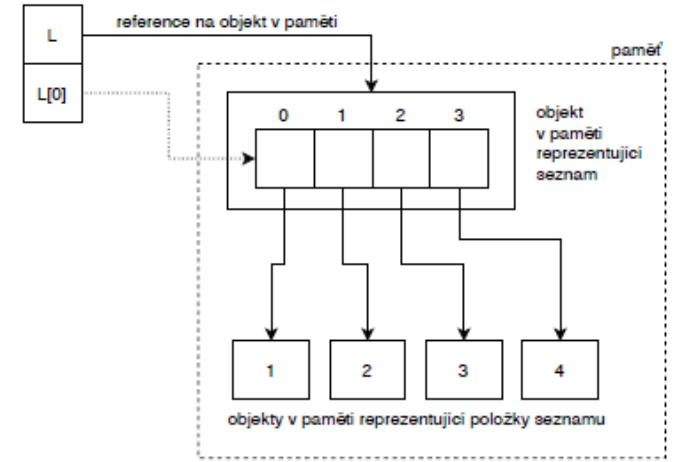
  L = ["spam", 2, 3, 4]

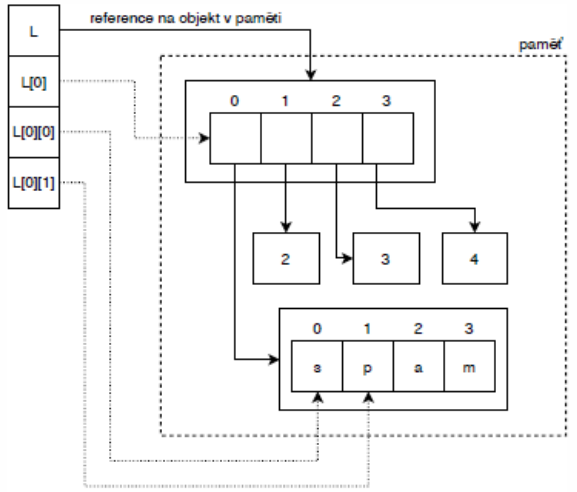
Práce se seznamy

- Pro práci se seznamy lze použít stejné nástroje jako pro práci s číselnými sekvencemi.
- Příklad:

```
L = [1, 2, 3, 4]
print(L[1]**L[2]) # vypíše: 8
L = ["spam", 2, 3, 4]
L = [[1, 2], 3, 4, 5]
```







Mutabilita seznamu

- Seznamy jsou na rozdíl od číselných sekvencí a řetězců mutabilní.
- Jejich jednotlivé položky lze měnit.
- Příklad:

```
L = [1, 2, 3, 4, 5]
L[0] = 42
print(L) # vypíše: [42, 2, 3, 4, 5]
```

- Mutabilita přináší drobnou komplikaci. Ke změně objektu může dojít skrze různé reference.
- Příklad:

```
L = [1, 2, 3, 4, 5]
M = L # M nyní obsahuje referenci na L
M[0] = 42 # změní referenci v L
print(L) # vypíše: [42, 2, 3, 4, 5]
```

- V následujícím kódu výše uvedené nenastává:

```
L = [1, 2, 3]
M = [4, 5]
N = [L, M]
M = 42
print(N) # vypíše: [[1, 2, 3], [4, 5]]
```

Metody pro práci se seznamy

- .append(objekt) přidá objekt na konec seznamu,
- .extend(sekvence) přidá všechny položky sekvence na konec seznamu,
- .remove(hodnota) odstraní ze seznamu první výskyt objektu s hodnotou hodnota,
- .pop() odstraní poslední prvek seznamu,
- .pop(index) odstraní prvek seznamu na daném indexu,
- .clear() odstraní všechny položky seznamu,
- .copy() vrací kopii seznamu (nový objekt),
- .reverse() vrací seznam s prvky v obráceném pořadí.
- .conut(hodnota) vrací počet prvků seznamu s hodnotou hodnota

Příklad

```
s=[]
for i in range(0,10):
    s.append(int(input("Zadej teplotu:")))

soucet=0
for i in s:
    soucet+=i

print(f"Průměrná teplota je {soucet/len(s):.2f}")

max=s[0]
for i in range(1,len(s)):
    if s[i]>max:
        max=s[i]

print(f"Nejvyšší teplota je {max}")
```

Úkol 1

- Napište program, který odstraní všechny výskyty zadané hodnoty ze seznamu.

- Řešení:

```
seznam=[1,5,4,7,8,9,1,10,5]

odstranit=5
i=0
while i <len(seznam):
    if seznam[i]==odstranit:
        seznam.pop(i)
    i+=1


print(seznam)
```

Úkol 1

- Řešení:

```
seznam=[1,5,4,7,8,9,1,10,5]

odstranit=5
i=0
while (seznam.count(odstranit)):
    seznam.remove(odstranit)

print(seznam)
```

Funkce

Příklad

- Vezměme si program, který počítá absolutní hodnotu čísla:

```
num = -4

if num < 0:
    num *= -1

print(num)
```

- Pokud chceme spočítat absolutní hodnotu pro jiné číslo, musíme kód znovu napsat.
- Chceme tedy náš kód vykonat vícekrát s různým vstupem.
- Za tímto účelem by bylo vhodné mít možnost si část programu pojmenovat, určit co jsou vstupní proměnné a co je výstupní hodnota.
- Právě k tomu slouží uživatelské funkce.

Funkce

- osamostatněné části programu
- „komunikují“ s jinými funkcemi prostřednictvím volání těchto funkcí
- při volání funkce dojde k provedení příkazů jejího těla
- parametry funkce = vstup funkce
- návratová hodnota = výstup funkce

Definice funkce

- syntaxe:

```
def jméno_funkce(parametr_1, parametr_2, ..., parametr_n):
    příkazy
```

  - `jméno_funkce` představuje identifikátor funkce (formálně se jedná o proměnnou, která obsahuje referenci na objekt reprezentující, danou funkci),
  - `parametr_1, . . . , parametr_n` jsou parametry funkce = proměnné, jež jsou přístupné v těle funkce,
  - tělo funkce, tvořené `příkazy`, je blokem kódu a musí být korektně odsazeno.

Příklad

```
def abs(num):
    if num < 0:
        return num * -1

def mocnina(x,n):
    return x**n
```

Volání funkce a návratová hodnota

- funkci voláme pomocí operátoru volání funkce () uvedeným za identifikátorem dané funkce; uvnitř závorek dále uvedeme skutečné parametry oddělené čárkou
- příklad:

```
mocnina(2,4)
```

- pro opuštění funkce s danou návratovou hodnotou slouží příkaz `return`
- není povinné mít vracet hodnotu z každého místa funkce, ale může to být zdroj chyb
- volání funkce může být součástí složitějšího výrazu, takto lze využít návratovou hodnotu dané funkce

Příklad

```
def abs(num):
    if num < 0:
        return num * -1
cislo=int(input("Zadej číslo:"))
cislo=abs(cislo)
print(cislo)
```

Příklad

```
def mocnina(x,n):
    return x**n

cislo=int(input("Zadej číslo:"))
n=int(input("Zadej n:"))
print(f"Mocnina: {mocnina(cislo,n)}")
```

Příklad

```
def tisk(s):
    print("Seznam obsahuje tato čísla: ");
    for i in s:
        print(f"{i},",end=" ")

def soucet(s):
    suma=0
    for i in s:
        suma+=i
    return suma

cisla=[8,4,5,6,7,1,2,9,10,24]
tisk(cisla)
print(f"Součet čísel v je: {soucet(cisla)}")
mocniny=[]
for i in cisla:
    mocniny.append(i**2)

tisk(mocniny)

print(f"Součet mocnin je: {soucet(mocniny)}")
```

Příklad

```
def obsah_obdelniku(s1, s2):
    if(s1>0 and s2>0):
        return s1*s2

a=int(input("Zadej stranu a:"))
b=int(input("Zadej stranu b:"))


print(f"Obsah obdelníku je: {obsah_obdelniku(a,b)}")
```

- Co když změníme hodnotu proměnné v na -8?

Příklad

```
def obsah_obdelniku(s1, s2):
    if(s1>0 and s2>0):
        return s1*s2
    else:
        return "Chybné zadání stran."

a=int(input("Zadej stranu a:"))
b=int(input("Zadej stranu b:"))


print(f"Obsah obdelníku je: {obsah_obdelniku(a,b)}")
```

Příklad

```
def obsah_obdelniku(s1, s2):
    if(s1>0 and s2>0):
        return s1*s2
    else:
        return "Chybné zadání stran."

def obsah_ctverce(s1):
    return obsah_obdelniku(s1,s1)

a=int(input("Zadej stranu a:"))
b=int(input("Zadej stranu b:"))


print(f"Obsah obdelníku je: {obsah_obdelniku(a,b)}")
print(f"Obsah čtverce je: {obsah_ctverce(a)}")
```

Přejmenování vestavěných identifikátorů

- vestavěné identifikátory lze používat jako vlastní identifikátory:

```
range=6
def input(x):
    return x
print (input(range))
```

- přejmenované identifikátory nelze použít v původním významu:

```
>>> a=range(1,10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
```

Polymorfismus funkce

- vytvoříme funkci:

```
>>> def times(x,y):
...     return x*y
...
```

- funkci zavoláme s celými čísly:

```
>>> print(times(3,5))
15
```

- ale i s číslem a řetězcem:

```
>>> print(times("ahoj",5))
```

- ale ne se dvěma řetězci (* není definován pro dva řetězce):

```
ahojahojahojahojahoj
>>> print(times("ahoj","světe"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in times
TypeError: can't multiply sequence by non-int of type 'str'
>>>
```

Příklad

```
def prunik(x,y):
    pr=[]
    for a in x:
        if a in y:
            pr.append(a)
    return pr

r=range(1,10)
s=[1,2,3,5,6]
t=[1,[2,3],5,6]
print(prunik(r,s))
[1, 2, 3, 5, 6]
print(prunik(s,r))
[1, 2, 3, 5, 6]
print(prunik(t,s))
[1, 5, 6]
print(prunik(t,"ahoj"))

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 4, in prunik
TypeError: 'in <string>' requires string as left operand, not int
```

Příklad

```
op=input("Zadej operaci:")
x=int(input("Zadej první číslo:"))
y=int(input("Zadej druhé číslo:"))

if op=="+":
    def f(x,y):
        return x+y

if op=="-":
    def f(x,y):
        return x-y

if op=="*":
    def f(x,y):
        return x*y

if op=="/":
    def f(x,y):
        return x/y

print(f"Výsledek operace {x}{op}{y} je {f(x,y)}.")
```

Úkol 2

- Napište funkci

  `xteprvky(seznam,x)`

  - která pro seznam s vrátí seznam jehož prvky budou prvky seznamu na místech s násobky čísla x.

- Řešení:

```
def xteprvky(seznam,x):
    s=[]
    i=x-1
    while i<len(seznam):
        s.append(seznam[i])
        i+=x
    return s

s1=[1,2,3,4,5,6,7,8,9,10]
s2=xteprvky(s1,2)

print(s2)
```

Úkol

Napište funkci odpovídající deklaraci

```
preved(znak)
```

která daný znak převede podle těchto pravidel:

- pokud je znak malé písmeno, převede ho na velké písmeno
- pokud je znak velká písmeno, převede ho na malé písmeno,
- pokud je znak číslo, zvýší ho o 5 modulo 10 (tedy 5 se změní na 0, 1 na 6)

Funkci pak vyzkoušejte v hlavním programu.

Příklad výstupu: