Laboratorul 4

Formatori:
Tutor: Militaru Mihai-Adrian
Tutor: Dragomir Titian-Cornel
+8

Data de începere a cursului:
25.09.2023
Utilizatori înscriși
Calendar
Note

Laboratorul 4

Liste in Python

Listele sunt un tip de date abstract în care elementele sunt stocate într-o manieră ordonată. În Python, o listă este creată prin adaugarea elementelor în paranteze drepte [], separate prin virgulă.

```
lista = [1, 2, 3]
print(lista)
```

Listele pot contine diferite tipuri de date: numere intregi, reale, caractere, liste, tuple.

```
#lista intregi
lista_int = [1, 2, 3]

#lista caractere
lista_char = ['a', 'b', 'c']

#lista mixta
lista_mix = [1, 'a', [2]]
```

Accesarea elementlor listei prin indecși

Putem folosi operatorul index [] pentru a accesa un element dintr-o listă. Indecșii încep de la 0, deci o listă cu 3 elemente va avea un index de la 0 la 2.

```
lista = [1, 2, 3, 4, 5]

#primul element
print(lista[0])

#al doilea element
print(lista[1])

#ultimul element
print(lsita[-1])

#penultimul element
print(lsita[-2])
```

Taierea unei liste (List Slicing)

Putem accesa o serie de elemente dintr-o listă folosind operatorul de tăiere : (două puncte) .

Când tăiem liste, indexul de început este inclusiv, iar indexul de final este exclusiv. De exemplu, lista[2: 4] returnează o listă cu elemente la indexul 2, 3, dar nu 4.

```
lista = [1, 2, 3, 4, 5]
print(lista[2:4])

>>[3, 4]
```

Capul, coada listei

Capul listei conține doar primul element al listei - accesat prin operatorul index []. Coada listei conține toate elementele rămase de la al doilea până la ultimul element de listă - accesat prin operația de tăiere.

```
lista = [1, 2, 3, 4, 5]
# capul (head)
print(lista[0])

#coada (tail)
print(lista[1:])
```

Concatenarea a doua liste

Putem folosi operatorul + pentru a combina două liste.

```
lista1 = [1, 2, 3, 4, 5]
lista2 = [6, 7]
concat = lista1 + lista2
print(concat)
```

Parcurgerea unei liste in mod recursiv

Nota: pentru rezolvarea exercitiilor de la acest laborator nu folositi structuri repetitive (for, while), folositi recursivitatea!

Ne folosim de notiunile de cap si coada prezentate anterior pentru a parcurge listele.

```
def afisare(lista):
    if len(lista) >= 1:
        cap = lista[0]  # primul element din lista
        coada = lista[1:]  #  toate elementele rămase de la al doilea până la ultimul element de listă
        print(cap)
        afisare(coada)

afisare([1, 2, 3, 4, 7])
```

Accesarea unui index care nu exista

```
lista = [1, 2, 3]
print(lista[7])

>> IndexError: list index out of range
```

Funcția reduce()

Funcția reduce() este definită în modulul functools. Funcția reduce() primește două argumente, o funcție și un iterabil (in cazul nostru, o lista) si returnează o singură valoare. Deasemenea, functia reduce() are un argument optional: o valoare initiala. Daca aceasta valoare initiala este prezenta, ea va fi plasata inaintea tuturor elementelor in calcul.

```
functools.reduce(functia, iterabil, valoare initiala)
```

Pentru a calcula suma tuturor intregilor dintr-o lista putem folosi reduce() impreuna cu o functie anonima (diefinita cu ajutorul lambda).

```
import functools

suma = functools.reduce(lambda a, b: a + b, [1, 2, 3])
print(suma)
```

Putem folosi si modulul operator, invatat in laboratorul 2.

```
import functools
import operator

suma = functools.reduce(operator.add, [1, 2, 3])
print(suma)
```

Funcția filter()

Funcția filter() primește, la fel ca functia reduce(), două argumente, o funcție și un iterabil. Insă nu returnează o singură valoare, ci returnează un alt iterabil. După cum îi spune și numele, functia creează o listă de elemente pentru care funcția returnează adevărat.

```
rezultat = list(filter(lambda x: x % 2 == 0, [1, 2, 3, 4, 5])) # numere pare
print(rezultat)
```

Funcția map()

La fel ca funcțiile filter și reduce, funcția map() primește două argumente, o funcție și un iterabil. Functia aplică funcția tuturor elementelor din lista.

```
rezultat = list(map(lambda x: x+1, [1, 2, 3, 4, 5]))
print(rezultat)
```

Metode pentru liste (funcții pe liste)

Python are un set de metode predefinite care permit lucrul cu liste:

```
append() #Adaugă un element la finalul listei

clear() #Șterge toate elementele din listă

copy() #Returnează o copie a listei primite ca parametru

extend() #Adaugă elementele unei liste la sfârșitul listei curente

index() #Returnează indexul primei apariții a elementului primit ca parametru

insert() #Adaugă un element la poziția specificată

pop() #Șterge elementul de pe poziția specificată

remove() #Șterge prima apariție a elementului dat ca parametru

reverse() #Inversează ordinea elementelor din listă

sort() #Sortează lista
```

◄ Curs 4 - Liste

Sari la...

Exerciții - Săptămâna 4 ►

Contactați serviciul de asistență

Sunteți conectat în calitate de Ciobanu Daria-Andreea (Delogare)
S1-L-AC-CTIRO1-LSD

Meniul meu