Funkcionalno programiranje

Funkcionalno programiranje

Od svakog funkcionalnog programskog jezika se očekuje da sledi ove koncepte:

- Čiste funkcije (**Pure Functions**): Ove funkcije imaju dva glavna svojstva. Prvo, oni uvek proizvode isti izlaz za iste argumente, bez obzira na bilo šta drugo. Drugo, nemaju nikakvih nuspojava.
- **Rekurzija**: Iteracija u funkcionalnim jezicima se sprovodi rekurzijom.
- Funkcije su prve klase (**First Class**) i mogu biti višeg reda (**Higher Order**): funkcije prve klase se tretiraju kao promenljive prve klase. Promenljive prve klase mogu se proslediti funkcijama kao parametar, mogu se vratiti iz funkcija ili sačuvati u strukturama podataka.
- **Promenljive su nepromenljive (immutable)**: U funkcionalnom programiranju ne možemo da menjamo promenljivu nakon što je inicijalizovana. Možemo da kreiramo nove promenljive - ali ne možemo da menjamo postojeće.

Funkcionalno programiranje u Python-u

-Čiste funkcije (**Pure Functions**)

Jedini rezultat rada čiste funkcije je povratna vrednost koju ona vraća.

```
    nova_lista = []

    for i in lista:
        nova_lista.append(i**2)

    return nova_lista


lista_org = [1, 2, 3, 4]
lista_izm = pure_func(lista_org)

print("Originalna Lista:", lista_org)
print("Izmenjena Lista:", lista_izm)

'''
Izlaz>>>

Originalna Lista: [1, 2, 3, 4]
Izmenjena Lista: [1, 4, 9, 16]

'''
```

Funkcionalno programiranje u Python-u

-**Rekurzija**

Rekurzija je proces u kojem se funkcija poziva direktno ili indirektno. U rekurzivnom programu daje se rešenje osnovnog slučaja, a rešenje većeg problema izražava se kroz manje probleme. Može se postaviti pitanje šta je osnovni slučaj? Osnovni slučaj se može smatrati uslovom koji govori kompajleru ili tumaču da izađe iz funkcije..

```
    # osnovni slučaj
    if n <= uslov:
        return count

    count += lista[uslov]

    # ulazak u rekurziju
    count = suma(lista, uslov+1, n, count)

    return count


lista = [1, 2, 3, 4, 5]
count = 0
n = len(lista)
print(suma(lista, 0, n, count))

'''
Izlaz>>>

15

'''
```

Funkcionalno programiranje u Python-u

-**First-Class | Higher-Order funkcije**

•Funkcija je instanca tipa Object.
•Funkciju možete sačuvati u promenljivoj.
•Možete preneti funkciju kao parametar drugoj funkciji.
•Možete vratiti funkciju iz funkcije.
•Možete ih skladištiti u strukturama podataka kao što su heš tabele, liste,…

Ugrađene funkcije višeg reda:
map(), filter(),

```
def shout(text):
    return text.upper()

def whisper(text):
    return text.lower()

def greet(func):
    # cuvanje funkcije u varijablu
    greeting = func("Zdravo, ja sam kreiran od strane funkcije koja je prosledjena kao argument "
    print(greeting)

greet(shout)
greet(whisper)

'''
Izlaz>>>
ZDRAVO, JA SAM KREIRAN OD STRANE FUNKCIJE KOJA JE PROSLEDJENA KAO ARGUMENT
zdravo, ja sam kreiran od strane funkcije koja je prosledjena kao argument
'''
```

Funkcionalno programiranje u Python-u

-**Lambda funkcije**

U Pythonu anonimna funkcija znači da je funkcija bez imena. Kao što već znamo, def se koristi za definisanje normalnih funkcija, a lambda za kreiranje anonimnih funkcija.

```
cube = lambda x: x * x*x
print(cube(7))


L = [1, 3, 2, 4, 5, 6]
is_even = [x for x in L if x % 2 == 0]

print(is_even)

'''
Izlaz>>>

343
[2, 4, 6]

'''
```

Funkcionalno programiranje u Python-u

-**Nepromenljivost varijabli**

Nepromenljivost u paradigmi funkcionalnog programiranja može se koristiti za otklanjanje grešaka jer će izbaciti grešku tamo gde se menja promenljiva, a ne tamo gde se menja vrednost. Python takođe podržava neke nepromenljive tipove podataka kao što su string, tuple, numerički itd.

```
kurs = "Advanced Python"
kurs[0] = "a"


'''
Izlaz>>>

Traceback (most recent call last):
  File "d:\Comtrade\Advanced Python\Cas 1\skripta.py",
  line 122, in <module>
    kurs[0] = "a"
TypeError: 'str' object does not support item assignment

'''
```


Hvala na pažnji
CODE
BY COMTRADE