NTP 2020/2021.

Vežba 1

U prvom terminu vežbi koje se tiču naprednih koncepata programskog jezika *Python*, studenti bi trebalo da se upoznaju sa sledećim tematskim celinama samostalno proučavajući dolenavedene materijale:

- Specijalne metode
  - poglavlje “Specijalne metode” (slajdovi 3-15)
  - modul specijalne_metode.py
- Properties
  - poglavlje “Properties” (slajdovi 16-19)
  - modul properties.py
- Deskriptori
  - poglavlje “Deskriptori” (slajdovi 20-24)
  - modul deskriptori.py
- Dekoratori
  - poglavlje “Dekoratori” (slajdovi 25-28)
  - modul dekoratori.py
- functools
  - poglavlje “functools” (slajdovi 29-36)
  - modul funkcije_viseg_reda.py
- itertools
  - poglavlje “itertools” (slajdovi 37-46)
    modul funkcije_za_iteraciju.py

Zadaci za samostalni rad

Nakon prolaska kroz gorenavedene materijale, ali i uz dodatno samostalno istraživanje, potrebno je uraditi sledeće zadatke (teorijske i praktične):

1. Šta je to *script* jezik? Navesti nekoliko primera.
2. Kada je nastao *Python* programski jezik? Ko je njegov tvorac?
3. Da li je *Python* statički ili dinamički tipiziran programski jezik?
4. Da li *Python* ima tipove podataka?
5. Navesti nekoliko najznačajnijih Python “implementacija”.
6. Ukratko objasniti pojam Just-In-Time kompajliranja?
7. Koja je referentna implementacija *Python* programskog jezik?
8. Da li je *Python* (*CPython*) interpretiran ili kompajliran programski jezik? Objasniti.
9. Šta je programska paradigma?
10. Šta znači da je *Python* multi-paradigmatski programski jezik?
11. Koje sve paradigme *Python* podržava?
12. Šta su i čemu služe magične metode? Navesti nekoliko primera.
13. Šta je iterator protokol (u kontekstu programskog jezika *Python*)? Navesti primer.
14. Koja je razlika između iteratora i generatora? Kada se koristi jedan, a kada drugi?
15. Koja je razlika između *__getattribute__* i *__getattr__*?
16. Implementirati klasu *PrirodniBrojevi* čiji je zadatak da omogući iteraciju kroz skup prirodnih brojeva. Implementirati ekvivalentni generator.

17. Šta predstavlja *__dict__*?
18. Da li u programskom jeziku *Python* postoje modifikatori pristupa (engl. *access modifiers*)?
19. Koje su dve prednosti deskriptora u odnosu na *properties*? Navesti primer.
20. Da li su funkcije objekti u programskom jeziku *Python*?
21. Da li su funkcije *first-class* objekti u programskom jeziku *Python*? Objasniti.
22. Šta su to unutrašnje (ugnježdene) funkcije?
23. Šta je leksičko zatvorenje?
24. Šta su dekoratori? Navesti nekoliko primera.
25. Da li dekoratori mogu da imaju parametre? Ukoliko mogu, navesti primer.
26. Šta su to *lamda* funkcije, a šta predikatske funkcije?
27. Koja dva modula u *Python* programskom jeziku omogućavaju paradigmu funkcionalnog programiranja?
28. Šta je to *lamda calculus*? Ko je njegov tvorac?
29. Koji je zadatak *@total_ordering* dekoratora? Navesti primer.
30. Čemu služi *@wraps dekorator*? Navesti primer.
31. U jednom iskazu potrebno je odrediti zbir kvadrata prvih 100 prirodnih brojeva. Napomena: Nije dozvoljena upotreba funkcije *sum*.
32. U jednom iskazu potrebno je odrediti zbir prvih 100 parnih prirodnih brojeva. Napomena: Nije dozvoljena upotreba funkcije *sum*.
33. Funkcija *faktorijel*[1] prima jedan parametar *n*, koji predstavlja prirodan broj. Napisati ovu funkciju tako da koristi samo jedan iskaz.
34. Funkcija *levi_faktorijel*[2] prima jedan parametar *n*, koji predstavlja prirodan broj. Napisati ovu funkciju tako da sadrži samo jedan iskaz. Potrebno je iskoristiti funkciju *faktorijel* iz prethodnog zadatka. Napomena: Nije dozvoljena upotreba funkcije *sum*.

1 https://en.wikipedia.org/wiki/Factorial
2 https://sh.wikipedia.org/wiki/Levi_faktorijel