**Python**

**ამოცანა 1.**

ქვემოთმოყვანილიდან რომელი მათგანი ამატებს fruits ლექსიკონს (Dictionary) ახალ წევრს, რომლის გასაღებიც (key) არის 'grapes'

და მნიშვნელობა (value) 15?

fruits = {'apples': 1, 'bananas': 4, 'pears': 17, 'oranges': 14}

A. fruits['grapes']

B. fruits['grapes'] = 15

C. insert 'grapes' in fruits

D. fruits[15] = 'grapes'

**ამოცანა 2.**

აირჩიეთ myfunc() ფუნქციის სწორი დეკლარაცია, რათა წარმატებით შევასრულოთ ფუნქციის გამოძახბები:

myfunc(2, 7, 67)

myfunc(10, 20)

A. def myfunc(\*\*kwargs)

B. არა, პითონში ეს შეუძლებელია

C. Def myfunc(args\*)

D. def myfunc(\*data)

**ამოცანა 3.**

შეადგინეთ პითონში ფუნქცია, რომელსაც პარამეტრად გადაეცემა სია (List) და აბრუნებს გადაცემული

უნიკალური ელემენტების ამორჩევის შედეგად მიღებულ სიას.

მაგ.

ფუნქციისათვის არგუმენტად გადაცემული სია : [1,2,3,3,3,3,4,5]

ფუნქციის მიერ დაბრუნებული სია : [1, 2, 3, 4, 5]

def unics(list):  
 temp\_list = []  
 for item in list:  
 if item in temp\_list:  
 continue  
 else:  
 temp\_list.append(item)  
 print(temp\_list)  
  
  
unics([1, 2, 3, 3, 3, 3, 4, 5])

**ამოცანა 4.**

ქვემოთ ჩამოთვლილთაგან რომელია სწორი?

A. პითონში ლექსიკონს(Dictionary) შეიძლება ჰქონდეს ორი ერთი და იგივე გასაღები (key) განსხვავებული მნიშვნელობებით.

B. პითონში ლექსიკონს(Dictionary) შეიძლება ჰქონდეს ორი ერთი და იგივე მნიშვნელობა (Value) სხვადასხვა გასაღებით (key)

C. პითონში ლექსიკონს(Dictionary) შეიძლება ჰქონდეს ორი ერთი და იგივე გასაღები ან ერთი და იგივე მნიშვნელობა, მაგრამ არ შეიძლება

ჰქონდეს ორი იგივე გასაღები-მნიშვნელობის წყვილი.

D. პითონში ლექსიკონს(Dictionary) არ შეიძლება ჰქონდეს ორი ერთი გასაღები (key) და არც ორი ერთი და იგივე მნიშვნელობა (Value).

**ამოცანა 5. (3 ქულა)**

მოცემული გაქვთ რიცხვებისაგან შემდგარი სია (List)

aList = [10, 20, 30, 40, 50, 60, 70, 80]

მოიყვანეთ ორი განსხვავებული მაგალითი აღნიშნული სიიდან ბოლო 2 ელემენტის პოვნის.

from itertools import islice  
  
  
def last\_two\_1(ls):  
 num\_elements = 2  
 print(ls[-num\_elements:])  
  
  
def last\_two\_2(ls):  
 res = list(islice(reversed(ls), 0, 2))  
 res.reverse()  
 print(res)  
  
  
last\_two\_1([10, 20, 30, 40, 50, 60, 70, 80])  
last\_two\_2([10, 20, 30, 40, 50, 60, 70, 80])

**ამოცანა 6.**

შექმენით კლასი ნებისმიერი სახელწოდებით და მოიყვანეთ კლასის დონის (Class Level) და

ობიექტის დონის (Instance level) ცვლადების მაგალითები, აღწერეთ რა სხვაობაა მათ შორის.

კლასის დონეზე ცვლადებს კლასის ცვლადებად მოიხსენიებენ, ხოლო ობიექტის დონეზე ცვლადებს ინსტანციის ცვლადებს უწოდებენ.ეს დიფერენციაცია საშუალებას გვაძლევს გამოვიყენოთ Class Level ცვლადები ობიექტების ინიციალიზაციისთვის ცვლადებისთვის მინიჭებული კონკრეტული მნიშვნელობით და გამოვიყენოთ სხვადასხვა ცვლადები თითოეული ობიექტისთვის Instance level ცვლადებით.

class Shark: # Class variables animal\_type = "fish" location = "ocean" # Constructor method with instance variables name and age def \_\_init\_\_(self, name, age): self.name = name self.age = age # Method with instance variable followers def set\_followers(self, followers): print("This user has " + str(followers) + " followers")

**ამოცანა 7.**

შექმენით კლასი Country, რომელსაც გააჩნია ორი წევრი ცვლადი CountryName და Countrypopulation. Country კლასს დაამატეთ კლასის

კონსტრუქტორი. შექმენით Country კლასის შვილობილი კლასი City, რომელსაც განუსაზღვრეთ წევრი ცვლადი CityName. City კლასს დაამატეთ

კლასის კონსტრუქტორი რომელსაც გადაეცემა სამი პარამეტრი CountryName , Countrypopulation და CityName და გამოიძახებს Country კლასის

კონსტრუქტორს (Parent Class Constructor)

class Country:  
  
 def \_\_init\_\_(self, country\_name, country\_population):  
 self.country\_name = country\_name  
 self.country\_population = country\_population  
  
  
class City(Country):  
  
 def \_\_init\_\_(self, country\_name, country\_population, city\_name):  
 Country.\_\_init\_\_(self, country\_name, country\_population)  
 self.city\_name = city\_name  
  
 def \_\_str\_\_(self):  
 return self.country\_name + " " + str(self.country\_population) + " " + self.city\_name  
  
  
city = City("geo", 22, "tbilisi")  
print(city)

**ამოცანა 8.**

ჩამოთვლილთაგან რომელია ინიციალზატორის/კონსტრუქტორის მეთოდის განსაზღვრის სწორი გზა?

A. def \_\_init\_\_(title, author):

B. def \_\_init\_\_(self, title, author):

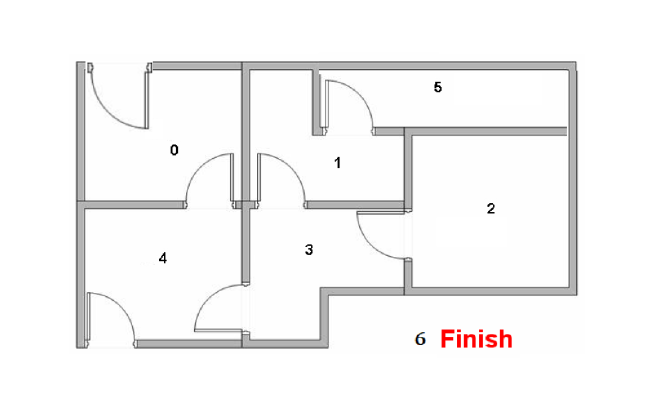
C. def \_\_init\_\_():

D. \_\_init\_\_(self, title, author):

**ამოცანა9.**

დაწერეთ Q-learning ალგორითმი პროგრამირების ენის Python-ის გამოყენებით, რომელიც დაეხმარება "აგენტს" იპოვოთ უმოკლესი მანძილი ლაბირინთში

მდებარე ნებისმიერი ოთახიდან ბოლო მე-6 ოთახამდე.



import numpy as np  
  
# R matrix 0 1 2 3 4 5 6  
R = np.matrix([[-1, -1, -1, -1, 0, -1, 100],  
 [-1, -1, -1, 0, -1, 0, -1],  
 [-1, -1, -1, 0, -1, -1, -1],  
 [-1, 0, 0, -1, 0, -1, -1],  
 [0, -1, -1, 0, -1, -1, 100],  
 [-1, 0, -1, -1, -1, -1, -1],  
 [0, -1, -1, -1, 0, -1, 100]  
 ])  
  
# Q matrix  
Q = np.matrix(np.zeros([7, 7]))  
  
# Gamma (learning parameter).  
gamma = 0.8  
  
# Initial state. (Usually to be chosen at random)  
initial\_state = 1  
  
  
# This function returns all available actions in the state given as an argument  
def available\_actions(state):  
 current\_state\_row = R[state,]  
 av\_act = np.where(current\_state\_row >= 0)[1]  
 return av\_act  
  
  
available\_act = available\_actions(initial\_state)  
  
  
def sample\_next\_action():  
 next\_action = int(np.random.choice(available\_act, 1))  
 return next\_action  
  
  
action = sample\_next\_action()  
  
max\_index = np.where(Q[action,] == np.max(Q[action,]))[1]  
  
  
def update(current\_state, action, gamma):  
 max\_index = np.where(Q[action,] == np.max(Q[action,]))[1]  
  
 if max\_index.shape[0] > 1:  
 max\_index = int(np.random.choice(max\_index, size=1))  
 else:  
 max\_index = int(max\_index)  
 max\_value = Q[action, max\_index]  
  
 # Q learning formula  
 Q[current\_state, action] = R[current\_state, action] + gamma \* max\_value  
  
  
# Update Q matrix  
update(initial\_state, action, gamma)  
  
for i in range(10000):  
 current\_state = np.random.randint(0, int(Q.shape[0]))  
 available\_act = available\_actions(current\_state)  
 action = sample\_next\_action()  
 update(current\_state, action, gamma)  
  
current\_state = 5  
steps = [current\_state]  
  
while current\_state != 6:  
  
 next\_step\_index = np.where(Q[current\_state,] == np.max(Q[current\_state,]))[1]  
  
 if next\_step\_index.shape[0] > 1:  
 next\_step\_index = int(np.random.choice(next\_step\_index, size=1))  
 else:  
 next\_step\_index = int(next\_step\_index)  
  
 steps.append(next\_step\_index)  
 current\_state = next\_step\_index  
  
# Print selected sequence of steps  
print("Selected path:")  
print(steps)