socrative

# Examen scris AP re feb 2022 30 Questions

**1.** Cum verificăm dacă un dicționar conține cheia "info"?

32/50 **A** if "info" in dictionary:

20/50 **B** dictionary['info'] == True

3/50 **C** del dictionary['info']

38/50 **D** if 'info' in dictionary:

**2.** Dacă x = 14 și y = 18, ce tip de date va reține variabila z = x + y ?

2/51 **A** float

49/51 **B** int

0/51 **C** string

0/51 **D** Instructiunea ridica exceptie de tip ValueError

**3.** Fie lista = ['a','b','c','d','e','f']. Cum va arăta aceasta după instructiunea lista.insert(0,'x')?

0/51 **A** ['a', 'b', 'c', 'd', 'e', 'f', 'x']

0/51 **B** ['a', 'x', 'b', 'c', 'd', 'e', 'f']

51/51 **C** ['x','a', 'b', 'c', 'd', 'e', 'f']

0/51 **D** Eroare de sintaxa

**4.** Cum aflam numarul de elemente ale unei liste lst?

50/50 **A** len(lst)

0/50 **B** lst.length()

0/50 **C** lst.size()

0/50 **D** sizeof(lst)

**5.** Care dintre urmatoarele sunt diferente intre functii introduse cu def si functii lambda?

46/50 **A** O functie lambda nu are nume

0/50 **B** O functie lambda poate lua un singur parametru

3/50 **C** O functie lambda nu poate lua 0 parametri

1/50 **D** O functie lambda nu poate folosi list comprehensions

**6.** Ce folosim pentru a implementa generatori?

1/50 **A** return

1/50 **B** lazy

42/50 **C** yield

6/50 **D** super

**7.** Ce metoda folosim sa convertim un obiect x intr-un string JSON?

1/50 **A** json.convert(x)

46/50 **B** json.dumps(x)

0/50 **C** json.json(x)

3/50 **D** json.dump(x)

**8.** Ce metoda speciala este apelata cand comparam cu == doua instante ale unei clase?

4/51 **A** isinstance()

10/51 **B** __init__

33/51 **C** __eq__

4/51 **D** super()

**9.** Care dintre urmatoarele secvente reprezinta antetul unei functii care ia un parametru x pentru care exista un type hint care spune ca x este o lista de intregi?

3/50 **A** def f(list[int] x):

1/50 **B** def f(x: List[str]):

45/50 **C** def f(x: List[int]):

1/50 **D** def f(x: list of str):

**10.** Care paradigma de rezolvare a problemelor este potrivita pentru a afisa toate solutiile unei probleme?

4/51 **A** Programare dinamica

43/51 **B** Backtracking

0/51 **C** Divide et impera

4/51 **D** Greedy

**11.** Care dintre urmatoarele probleme sunt potrivite pentru a fi rezolvate folosind Greedy?

4/51 **A** Afisarea tuturor permutarilor unei multimi de n elemente

0/51 **B** Calcularea factorialului unui numar

4/51 **C** Calcularea drumului de cost maxim intr-o matrice in care ne putem deplasa doar o celula in jos sau o celula in dreapta la fiecare pas

43/51 **D** Problema planificarii spectacolelor

**12.** In secventa de cod din imagine, ce este x?

```
def f(x):
    x['key'] = 10
x = {'key': 20}
f(x)
print(x)
```

50/51 **A** Un dictionar

1/51 **B** O lista

0/51 **C** Un Set

0/51 **D** Codul nu este cod Python valid

**13.** In secventa de cod din imagine, f(x):

```
def f(x):
    x = {'key': 10}
x = {'key': 20}
f(x)
print(x)
```

12/51 **A** Va face un deepcopy a lui x

26/51 **B** Va face o copie a lui x

3/51 **C** Va da o eroare

10/51 **D** Va avea un comportament nedefinit

**14.** Care dintre urmatoarele variante completeaza corect spatiile libere astfel incat functia de mai jos sa returneze factorialul numarului dat ca parametru?

```
def factorial(n):
    result = 1
    for i in ____(2, ____):
        result = result __ i
    return result
```

7/50 **A** range, n, *

42/50 **B** range, n+1, *

0/50 **C** range, n+1, +

1/50 **D** list, n+1, *

**15.** Care dintre urmatoarele variante completeaza corect spatiile libere astfel incat functia de mai jos sa sorteze crescator lista data ca parametru?

```
def sort(lst):
    if lst == []:
        return []
    m = lst[0]
    a = [x for x in lst if ____]
    b = [x for x in lst if ____]
    c = [x for x in lst if ____]

    return sort(a) + b + sort(c)
```

44/51 **A** x < m, x == m, x > m

42/51 **B** x - m < 0, x - m == 0, x - m > 0

3/51 **C** x // m <= 1, x // m == 1, x // m > 1

7/51 **D** x <= m, x == m, x >= m

**16.** Care dintre urmatoarele variante reprezinta propietati ale seturilor (Sets) in Python?

47/50 **A** Sunt neordonate (unordered)

1/50 **B** Sunt ordonate

1/50 **C** Accepta duplicate

39/50 **D** Pot contine doar elemente imutabile

**17.** Care dintre urmatoarele afirmatii sunt adevarate in legatura cu seturile (Sets)?

46/50 **A** Nu accepta duplicate

46/50 **B** Pot contine tipuri de date diferite

4/50 **C** Nu pot contine tipuri de date diferite

6/50 **D** Obiectele retinute in ele pot fi modificate

**18.** Se defineste urmatoarea lista:
lst = list(range(100))

Care este rezultatul instructiunii asupra listei?
del lst[10:]

37/51 **A** lst va contine elementele 0, 1, 2, ..., 9

8/51 **B** lst va contine elementele 10, 11, 12, ..., 99

2/51 **C** lst va contine elementele 11, 12, 13, ..., 99

4/51 **D** Exceptie sau niciuna dintre variante

**19.** Care functie lambda este echivalenta functiei 'foo' de mai jos ?

```
def foo(x, y):
    c = x + y
    return c / 5
```

1/50 **A** lambda x, y: return (x+y) / 5

3/50 **B** lambda x, y: c = x + y; return c / 5

46/50 **C** lambda x, y: (x + y) / 5

0/50 **D** lambda x, y: c = x + y; c / 5

**20.** Care din urmatoarele instructiuni inlocuieste elementul 'are' cu stringul 'avea' in tuplul de mai jos: t = ('ana', 'are', 'mere')

3/51 **A** t[1] = 'avea'

0/51 **B** t[1:1] = 'avea'

17/51 **C** t = ('ana', 'avea', 'mere')

31/51 **D** Niciuna din variante

**21.** Care este modalitatea prin care se poate apela o metoda din clasa mostenita?

11/50 **A** self.__metoda()

36/50 **B** super().metoda()

1/50 **C** base().metoda()

2/50 **D** Nu este posibil

**22.** Care dintre urmatoarele variante completeaza corect spatiile libere astfel incat codul urmator sa afiseze [11,2022,13]?

```
import deepcopy

lst1=[11,12,13]
...
print(lst1)
```

10/51 **A**
```
lst1[1]=200
lst2=deepcopy(lst1)
lst2[1]=2022
```

34/51 **B**
```
lst2 = lst1
lst2[1]=2022
```

3/51 **C**
```
lst2 = lst1[:]
lst2[1]=2022
```

4/51 **D**
```
lst1[1]=2022
lst2=lst1
lst2[1]=300
```

**23.** Care este output-ul urmatoarei secvente de cod?

```
note = {'Ana': 8.5, 'Robert': 9.7}

if note['Ana'] > 9.5:
    print('Premiul I')
elif note['Robert'] > 9.5:
    print('Premiul I')
elif note['Alex'] > 9.5:
    print('Premiul I')
else:
    print('Punctaje prea mici')
```

5/51 **A** Se arunca exceptia 'KeyError' deoarece cheia 'Alex' nu exista in dictionar

45/51 **B** 'Premiul I'

0/51 **C** 'Punctaje prea mici'

1/51 **D** Eroare de sintaxa

**24.** Se da urmatoarea secventa de cod. Alegeti afirmatiile adevarate.

```
class Masina:
    def start(self):
        print('*motor pornit*')

    def self_park(self):
        print('*parcheaza*')

class Audi(Masina):
    def self_drive(self):
        print('*deplasare inainte*')

r8 = Audi()
```

2/51 **A** Codul va genera: AttributeError: 'Audi' object has no attribute 'start'

11/51 **B** Se va afisa: *deplasare inainte*

20/51 **C** Se va afisa: *motor pornit*

41/51 **D** Clasa Masina este clasa de baza pentru clasa Audi.

**25.** Care dintre urmatoarele variante completeaza corect spatiile libere astfel incat codul urmator sa nu arunce o exceptie?

```
import math

try:
    lst1 = [1, -4, 5]
    lst2 = [math.sqrt(x) for x in lst1]
...
```

1/51 **A** except KeyError: assert True except Exception: assert False

49/51 **B** except ValueError: assert True except Exception: assert False

19/51 **C**
```
except KeyError:
    assert False
except ValueError:
    assert True
```

38/51 **D** except KeyError: assert False except Exception: assert True

**26.** Care dintre urmatoarele variante completeaza corect spatiile punctate astfel incat codul urmator sa afiseze [100,2,3]?

```
def f(lst):
    ...

lst=[1,2,3]
f(lst)
print(lst)
```

30/51 **A** lst[:]=[100,2,3]

47/51 **B** lst[0]=100

49/51 **C**
```
lst.clear()
lst.append(100)
lst.append(2)
lst.append(3)
```

38/51 **D** lst.remove(1) lst.insert(0,100)

**27.** Se da urmatoarea secventa de cod:

```
class Masina:
    def start(self):
        print('*motor pornit*')

    def self_park(self):
        print('*parcheaza*')

class Audi(Masina):
    def self_drive(self):
        print('*deplasare inainte*')

class Ford(Masina):
    pass

class Mustang(Ford):
    pass

r8 = Audi()
focus = Ford()
mustang = Mustang()
```

Care dintre instructiuni vor fi evaluate la valoarea True?

47/50 **A** isinstance(r8, Masina)

22/50 **B** not isinstance(r8, Ford)

38/50 **C** isinstance(mustang, Masina)

41/50 **D** isinstance(focus, Ford)

**28.** Ce se va afisa in urma executiei secventei de cod?

```
class Masina:
    def start(self):
        print('*motor pornit*')

    def self_park(self):
        print('*parcheaza*')

class Ford(Masina):
    def self_drive(self):
        print('*deplasare inainte*')

class Mustang(Ford):
    def self_drive(self):
        super().self_drive()
        print('*accelerare*')

mustang = Mustang()
mustang.start()
mustang.self_drive()
```

0/50 **A** *motor pornit*

43/50 **B** *motor pornit* *deplasare inainte* *accelerare*

4/50 **C** *motor pornit* *accelerare*

3/50 **D** AttributeError: 'Mustang' object has no attribute 'start'

**29.** Fie un algoritm de complexitate timp ThetaMare(n). Căror clase de complexitate va aparține acesta?

41/51 **A** O(n)

23/51 **B** O(n*n*n)

35/51 **C** OmegaMare(log n)

17/51 **D** O(radical din n)

**30.** Fie funcția recursiva din imagine:

```
def aduna(x, val):
  if x<=0:
    return 0
  else:
    return aduna(x - 1, x + val)
```

Care afirmații sunt adevărate?

5/51 **A** Funcția returnează suma 1+2+3+...+n

42/51 **B** Funcția returnează 0 pentru orice n și val întregi

45/51 **C** Funcția are complexitatea timp O(n)

23/51 **D** Funcția va da eroare daca este apelata cu x intreg > 0 si val string