

1. Cum vom sterge cheia "model" dintr-un dictionar?

- 2/179 ☐ A delete model
- 3/179 ☐ B remove dict.model
- 146/179 ☒ C dict.pop("model")
- 158/179 ☐ D del dict['model']

2. Dacă $x = 14$ și $y = 18.25$, ce tip de date va reține variabila $z = x + y$?

- 177/179 ☒ A float
- 0/179 ☐ B int
- 1/179 ☐ C string
- 1/179 ☐ D Instructiunea ridica exceptie de tip ValueError

3. Fie lista = ['a','b','c','d','e','f']. Cum va arăta aceasta după instructiunea lista.insert(1,'x')?

- 1/179 ☐ A ['a', 'b', 'c', 'd', 'e', 'f', 'x']
- 171/179 ☒ B ['a', 'x', 'b', 'c', 'd', 'e', 'f']
- 1/179 ☐ C ['x','a', 'b', 'c', 'd', 'e', 'f']
- 6/179 ☐ D Eroare de sintaxa

4. Cati parametri poate lua o functie in Python?

- 0/180 ☐ A exact 1
- 0/180 ☐ B maxim 2
- 174/180 ☒ C oricati
- 6/180 ☐ D cel putin 1

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

- 154/181 ☒ A O functie lambda nu are nume
- 0/181 ☐ B O functie lambda poate lua un singur parametru
- 17/181 ☐ C O functie lambda nu poate lua 0 parametri
- 10/181 ☐ D O functie lambda nu poate folosi list comprehensions

6. Ce metoda folosim sa convertim un string x care contine JSON valid la obiect Python?

- 165/179 ☒ A json.loads(x)
9/179 ☐ B json.convert(x)
2/179 ☐ C json.to_python(x)
3/179 ☐ D json.to_obj(x)

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

- 9/179 ☐ A json.convert(x)
153/179 ☒ B json.dumps(x)
3/179 ☐ C json.json(x)
14/179 ☐ D json.dump(x)

8. Ce metoda speciala este apelata la instantierea unei clase?

- 3/180 ☐ A isinstance()
167/180 ☒ B __init__
0/180 ☐ C __eq__
10/180 ☐ 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 stringuri?

- 4/180 ☐ A def f(list[str] x):
164/180 ☒ B def f(x: List[str]):
3/180 ☐ C def f(x = List[str]):
9/180 ☐ D def f(x: list of str):

10. Care paradigma de rezolvare a problemelor presupune existenta subproblemelor suprapuse si a substructurii optime?

- 140/180 ☒ A Programare dinamica
5/180 ☐ B Backtracking
30/180 ☐ C Divide et impera
5/180 ☐ D Greedy

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

- 4/180 **A** Afisarea tuturor permutarilor unei multimii de n elemente
2/180 **B** Calcularea factorialului unui numar
21/180 **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
153/180 **D** Problema planificarii spectacolelor

12. Ce va afisa urmatoarea secventa de cod?

- 4/181 **A** 10
3/181 **B** 20
7/181 **C** KeyError
167/181 **D** {'key': 10}

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

13. Ce va afisa urmatoarea secventa de cod?

- 3/179 **A** 10
2/179 **B** 20
9/179 **C** KeyError
165/179 **D** {'key': 20}

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

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

- 23/179 **A** range, n, *
148/179 **B** range, n+1, *
4/179 **C** range, n+1, +
4/179 **D** list, n+1, *

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

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

- 157/179 **A** $x < m$, $x == m$, $x > m$
133/179 **B** $x - m < 0$, $x - m == 0$, $x - m > 0$
17/179 **C** $x // m <= 1$, $x // m == 1$, $x // m > 1$
35/179 **D** $x <= m$, $x == m$, $x >= m$

```
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)
```

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

- 163/180 **A** Sunt neordonate (unordered)
15/180 **B** Sunt ordonate
14/180 **C** Accepta duplicate
137/180 **D** Nu pot fi modificate (unchangeable / immutable)

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

- 157/179 **A** Nu accepta duplicate
133/179 **B** Pot contine tipuri de date diferite
35/179 **C** Nu pot contine tipuri de date diferite
59/179 **D** Datele pot fi modificate

18. Se defineste urmatorul dictionar: `d = {'a': 1, 'b': 2, 'c': 3}` Care este rezultatul instructiunii:
`d['b':'c']`

- 7/179 **A** [2, 3]
4/179 **B** (2, 3)
0/179 **C** 44230
168/179 **D** Exceptie

19. Care functie lambda este echivalenta functiei 'foo' de mai jos ?
`def foo(x, y): c = x + y return c / 5`

- 6/179 **A** `lambda x, y: return (x+y) / 5`
3/179 **B** `lambda x, y: c = x + y; return c / 5`
166/179 **C** `lambda x, y: (x + y) / 5`
4/179 **D** `lambda x, y: c = x + y; c / 5`

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

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

- 17/180 **A** `t[1] = 'avea'`
1/180 **B** `t[1:1] = 'avea'`
1/180 **C** `t(1) = 'avea'`
161/180 **D** Niciuna din variante

21. Care este modalitatea prin care se poate defini un atribut privat al unei clase, in Python?

- 169/179 **A** self.__foo
4/179 **B** private self.foo
0/179 **C** protected self.foo
6/179 **D** self.foo

22. Care dintre urmatoarele variante completeaza corect spatiile libere astfel incat codul urmator sa afiseze [11,200,13]?

- 149/181 **A** lst1[1]=200 lst2=deepcopy(lst1) lst2[1]=100
167/181 **B** lst2 = lst1 lst2[1]=200
18/181 **C** lst2 = lst1[:] lst2[1]=200
11/181 **D** lst1[1]=200 lst2=lst1 lst2[1]=300

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

- 22/181 **A** Se arunca exceptia 'KeyError' deoarece cheia 'Alex' nu exista in dictionar
155/181 **B** 'Premiul I'
1/181 **C** 'Punctaje prea mici'
3/181 **D** Eroare de sintaxa

24. Se da urmatoarea secventa de cod:

Ce se va afisa in urma executarii instructiunii: r8.start()?

- 15/179 **A** AttributeError: 'Audi' object has no attribute 'start'
6/179 **B** *deplasare inainte*
129/179 **C** *motor pornit*
29/179 **D** Nu se afiseaza nimic

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

- 28/180 **A** except KeyError: assert True except Exception: assert False
160/180 **B** except ValueError: assert True except Exception: assert False
43/180 **C** except Exception: assert False except ValueError: assert True
103/180 **D** except KeyError: assert False except Exception: assert True

```
import deepcopy
```

```
lst1=[11,12,13]
```

```
...
```

```
print(lst1)
```

```
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')
```

```
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()
```

```
import math
```

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

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

- 59/180 **A** lst=[100,2,3]
- 165/180 **B** lst[0]=100
- 166/180 **C** lst.clear()
lst.append(100)
lst.append(2)
lst.append(3)
- 133/180 **D** lst.remove(1) lst.insert(0,100)

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

```
...
```

```
lst=[1,2,3]
```

```
f(lst)
```

```
print(lst)
```

27. Se da urmatoarea secventa de cod:

Care dintre instructiuni vor fi evaluate la valoarea True?

- 149/179 **A** isinstance(r8, Masina)
- 23/179 **B** isinstance(r8, Ford)
- 131/179 **C** isinstance(mustang, Masina)
- 143/179 **D** isinstance(focus, Ford)

```
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()
```

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

- 2/179 **A** *motor pornit*
- 143/179 **B** *motor pornit* *deplasare inainte* *accelerare*
- 15/179 **C** *motor pornit* *accelerare*
- 19/179 **D** AttributeError: 'Mustang' object has no attribute 'start'

```
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()
```

29. Fie un algoritm de complexitate timp $O(n \log n)$ și $\Omega(n \log n)$. Căror clase de complexitate va aparține acesta?

- 54/181 **A** $O(n)$
- 160/181 **B** $\Theta(n \log n)$
- 113/181 **C** $\Omega(n)$
- 137/181 **D** $O(n^2)$

30. Fie funcția recursivă din imagine:

Care afirmații sunt adevărate?

- 32/179** ☐ **A** Funcția returnează suma $1+2+3+\dots+n$
- 142/179** ☒ **B** Funcția returnează 0 pentru orice n și val întregi
- 157/179** ☒ **C** Funcția are complexitatea timp $O(n)$
- 8/179** ☐ **D** Funcția are o eroare de sintaxă

```
def aduna(x, val):
```

```
    if x<=0:
```

```
        return 0
```

```
    else:
```

```
        return aduna(x - 1, x + val)
```

Care afirmații sunt adevărate?