Test Trial Midterm - Introduction to Python - 23/24

Question 1 – Basics–Logic–1 – 354642.2.1

Consider the following code snippet:

```
a = True
b = False
c = True
d = False

if not (a or b) and (c or d):
    print("True")
else:
    print("False")
```

What will be printed by the code?

A `True`

B `False`

C The code will result in an error.

D `None`

Question 2 – Basics–Logic–2 – 354643.1.1

Assume you have the following variables:

```
x = 3
y = 3.0
z = '3'
```

What will be printed by the following code:

```
print((x != z and x == int(z)) or (x != y and y != z))
```

A `True`

B `False`

C The code will result in an error.

D `3`

Question 3 − Basics−Loop−1 − 354644.1.2

Consider the following code snippet:

```
i = 1

while True:
    if i % 2 == 0:
        i += 1
        continue
    elif i == 7:
        break
    else:
        print(i)
        i += 1
```

Which of the following code snippets will print the same output?

A

```
for i in range(1, 5, 2):
    print(i)
```

B

```
for i in range(7, 1, -2):
    print(i)
```

C

```
x = 'numbers'
for i, j in enumerate(x):
    if i % 2 != 0:
        print(i)
```

D

```
x = '12345'
for i in x:
    if i % 2 != 0:
        print(i)
```

Question 4 − Basics−Loop−3 − 355236.1.3

Consider the following code snippet:

```
a = 1
b = 2.0
c = '3'
d = 'hello'
y = {}

for i in [0,1]:
    if i==0:
        if type(a)==type(b):
            y[i] = True
        else:
            y[i] = False
    if i==1:
        if type(c)==type(d):
            y[i] = True
        else:
            y[i] = False
print(y)
```

What will this code snippet print?

A `{0: False, 1: True}`

B `{0: True, 1: True}`

C `{0:  0, 1: 1}`

D `{0:  1, 1: 1}`

Question 5 − Basics−Scope−1 − 354687.1.2

Which other program will produce the same output as the following code snippet?

```
num = 300

def main():
  num = 2000
  print(num)

main()
print(num)
```

A

```
def main():
  global num
  num = 300

main()
print(num)
```

B

```
num = 300

def main():
  global num
  num = 2000

main()
print(num)
```

C

```
def outer():
    num = 2000
    def inner():
        num = 300
        print(num)
    print(num)

outer()
inner()
```

D

```
num = 300

def outer():
    num = 2000
    def inner():
        num = 300
        print(num)
    print(num)

outer()
print(num)
```

Question 6 – Basics–Variables–2 – 354641.1.3

Consider the following code snippet:

```
x = 10
y = '5'
z = x + y
```

What will be the value of '`z`' and why?

A The value of `z` will be `15` because the string `'5'` will be implicitly converted to an integer.

B The value of `z` will be `'105'` because Python concatenates strings.

C The code will result in an error because you can't add an integer and a string together.

D The code will result in a None because 'x' and 'y' have different types.

Question 7 − dictionaries − function − 1 − 354875.2.5

Suppose you have the following a dictionary called `students`, which contains three sub-dictionaries:

```
students = {"james": {"name": "James",
                      "homework": [90.0, 97.0, 75.0, 92.0],
                      "quizzes": [88.0, 40.0, 94.0],
                      "tests": [75.0, 90.0]},
            "eva":   {"name": "Eva",
                      "homework": [100.0, 92.0, 98.0, 100.0],
                      "quizzes": [82.0, 83.0, 91.0],
                      "tests": [89.0, 97.0]},
            "david": {"name": "David",
                      "homework": [0.0, 87.0, 75.0, 22.0],
                      "quizzes": [0.0, 75.0, 78.0],
                      "tests": [100.0, 100.0]}
            }
```

You want to create a function that calculates the average final course grade of a student.

The final course grade is calculated as 20% average homework grade, 10% average quizzes grade and 70% average tests grade.

Which of the following programs will work as intended and will give the final course grade of Eva?

A

```
def average(numbers):
    return sum(numbers)/(len(numbers))

def final_average(student):
    homework = average(student['homework'])
    quizzes = average(student['quizzes'])
    tests = average(student['tests'])
    return round(0.2*homework + 0.1*quizzes + 0.7*tests, 2)

final_average(students["eva"])
```

B

```
def final_average(student):
    def average(numbers):
        return sum(numbers)/(len(numbers)
    homework = average(student['homework'])
    quizzes = average(student['quizzes'])
    tests = average(student['tests'])
    return round(0.2*homework + 0.1*quizzes + 0.7*tests, 2)

final_average(students["eva"])
```

C Both functions will give the correct output.

D Neither functions will give the correct output.

Question 8 − dictionaries − logic − 1.2 − 354608.1.6

You need to write a function called `dict_keys` that checks whether a certain key is in available in a certain dictionary.

For example:
If we call the function as `dict_keys({"name": "Josh", "age": 56}, 'city')`
it should return `False`.

If we call the function as `dict_keys({"name": "Josh", "age": 56}, 'name')`
it should return `True`.

Which of the following code snippets will work as intended?

A

```
def dict_keys(dictionary, key):
   if key in dictionary.keys():
     return True
   else:
     return False
```

B

```
def dict_keys(dictionary, key):
   if key not in dictionary:
     return True
   else:
     return False
```

C

```
def dict_keys(dictionary, key):
   if not(key in dictionary.keys()):
     return True
   else:
     return False
```

D

```
def dict_keys(dictionary, key):
   if key in dictionary.items():
     return True
   else:
     return False
```

Question 9 – dictionaries – loop (enumerate) – 1 – 354857.1.13

Suppose you have a dictionary called `characters`, of which keys are strings and values are lists of strings.

```
characters = {'first' : ['a', 'b', '1', 'c', '5'],
              'second': ['d', '2', 'e', '6', 'f'],
              'third' : ['3', 'g', '7', 'h', 'i']}
```

You want to modify the dictionary such that all the elements in the value lists are only character strings and not numeric strings.
The intended output should look like this:

```
characters = {'first':  ['a', 'b', 'c'],
              'second': ['d', 'e', 'f'],
              'third':  ['g', 'h', 'i']}
```

Which of the following programs will work as intended?

Hints:
The `remove()` list method removes the first occurrence of the element with the specified value.
The `isnumeric()` string method returns `True` if all the characters are numeric (0-9), otherwise `False`.

A

```
for array in characters.values():
    for element in array:
        if element.isnumeric():
            array.remove(element)
```

B

```
for array in characters.values():
    for index, element in enumerate(array):
        if element.isnumeric():
            del array[index]
```

C

```
for key, value in characters.items():
    for element in value:
        if element.isnumeric():
            del value[value.index(element)]
```

D All of the programs will work as intended.

Question 10 – dictionaries – straightforward – 1.2 – 355315.2.4

Suppose you have the following dictionary:

```
library = {"books": 5,
           "genres": ['action', 'romance', 'thriller', 'classics']}
```

You want to:
- add a key "`years`", of which the value is a list of strings that are the years of publication, i.e., `["1932" , "2012", "1899", "2000", "1984"]`,
- change the value of the key "`books`" to `6`.

The new value of the dictionary `library` should be:

```
{"books" : 6,
 "genres" : ['action', 'romance', 'thriller', 'classics'],
 "years" : ["1932" , "2012", "1899", "2000", "1984"]}
```

Which of the programs below will work as intended?

A

```
library["books"] = 6
library.update({"years": ["1932" , "2012", "1899", "2000", "1984"]})
```

B

```
library["books"] = 6
library["years"] = ["1932" , "2012", "1899", "2000", "1984"]
```

C Both of the given options

D None of the given options

Question 11 − dictionaries − straightforward − 2.1 − 354900.1.5

Suppose you have the following function:

```
def my_count_function(string):
    counts = {}
    for letter in string:
        if letter in counts:
            counts[letter] += 1
        else:
            counts[letter] = 1
    return counts
```

And you create the following two dictionaries:

```
apple_count = my_count_function("apple")
peach_count = my_count_function("peach")
```

Which of the code snippets of code below prints something different than the other code snippets?

A `print(sum(apple_count.values()))`

B `print(len(apple_count.keys()))`

C `print(sum(peach_count.values()))`

D `print(len(peach_count.keys()))`

Question 12 − dictionaries − straightforward − 3.1 − 354898.1.4

Suppose you have the following dictionary:

```
x = {1: 10, 2: 20, 3: 30, 4: 40}
```

What will the following line of code print?

```
print(list(x.values()) + list(x.keys()))
```

A `[10, 20, 30, 40, 1, 2, 3, 4]`

B `[11, 22, 33, 44]`

C This code will result in an error because you cannot concatenate two lists using the + operator.

D `[(1, 10), (2, 20), (3, 30), (4, 40)]`

Question 13 − Function−Argument−1 − 355258.2.3

You have to write a function called `summarize` which take a list of integers as an argument.

The function should return a dictionary with the following key-value pairs:

- `amount`: the number of the integers from the input list
- `smallest`: the smallest integer from the input list
- `largest`: the largest integer from the input list
- `total`: the sum of all integers from the input list

Which of the following functions will achieve what you need?

A

```
def summarize(x):

    y = {}

    x['amount'] = len(y)
    x['smallest'] = min(y)
    x['largest'] = max(y)
    x['total'] = sum(y)

    return x
```

B

```
def summarize(x):
    y = {}
    total = 0

    for integer in x:
        y['amount'] = len(x)

        if integer == min(x):
            y['smallest'] = min(x)
        elif integer == max(x):
            y['largest'] = max(x)
        else:
            y['total'] += integer

    return y
```

C

```
def summarize(x):
    y = {}

    for integer in x:
        if integer == min(x):
            y['smallest'] == min(x)
        elif integer == max(x):
            y['largest'] == max(x)
        else:
            y['amount'] == x.count()
            y['total'] == sum(x)

    return y
```

D None of the functions will work as needed.

Question 14 – Function–Argument–Flexible–1 – 355280.2.5

Suppose you have the following function:

```
def multiply(*args, factor=2):
    total = 1
    for num in args:
        total *= num
    return total * factor
```

Which of the following functions will return the same values as the function above, if we call the function thrice as follows:

```
multiply(1, 2, factor=3)
multiply(3, 4)
multiply()
```

A

```
def multiply(*args, factor=2):
    total = args[0]
    for num in args:
        total *= num
    return total * factor
```

B

```
def multiply(*args, factor=2):
    if not args:
        return 0
    total = args[0]
    for num in args[1:]:
        total *= num
    return total * factor
```

C Both functions will return the same values.

D Neither functions will return the same values.

Question 15 – Function–Lambda–2 – 355319.2.3

Consider the following code snippet:

```
def main(lst, condition=lambda x: x):
    y = []
    for x in lst:
        if condition(x):
            y.append(x)
    return y
```

What will this function return, when called in the following way:

```
main([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], lambda x: x % 2 != 0)
```

A `[1, 3, 5, 7, 9]`

B `[2, 4, 6, 8, 10]`

C `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`

D `None`

Question 16 – Function–Logic–2 – 354678.1.8

You need to write a function called `weather_alert` that accepts a string argument, which is a color code of the weather alert system.

Your function should return one of the following lines to inform the citizens of what to do depending on the color:
- If the color is `red`, return: "Take action to protect against severe condition."
- If the color is `orange`, return: "Be prepared for damages."
- If the color is yellow, return: "Be alert for potential hazard."
- For any other colors, return: "No warning."

Which of the following program will do want you need?

A
```
def weather_alert(color):
    if color == "red":
        return "Take action to protect against severe condition."
    elif color == "orange":
        return "Be prepared for damages."
    elif color == "yellow":
        return "Be alert for potential hazard."
    return "No warning."
```

B
```
def weather_alert(color):
    if color == "red":
        print("Take action to protect against severe condition.")
    elif color == "orange":
        print("Be prepared for damages.")
    elif color == "yellow":
        print("Be alert for potential hazard.")
    else:
        print("No warning.")
```

C
```
def weather_alert(color):
    if color == "red":
        return "Take action to protect against severe condition."
    elif color == "orange":
        return "Be prepared for damages."
    elif color == "yellow":
        return "Be alert for potential hazard."
    elif color == "green":
        return "No warning."
```

D All of the programs should work as intended.

Question 17 – Function–Loop–1 – 355249.2.3

Suppose you have the following function which take a list of numerical strings as an argument.

```
def counter(y):
    x = {}
    for index, element in enumerate(y):
        if index != int(element) and int(element) % 2 == 0:
            x[element] = len(element)
    return x
```

What will this function return, when we call the function as follows:
`counter(['12345678', '1', '123', '1234', '12345', '12', '6', '1', '123456'])`

A `{'12345678': 8, '1234': 4, '12': 2, '123456': 6}`

B `{'12345678': 8, '123': 3, '1234': 4, '12345': 5, '12': 2, '6': 1, '1': 1, '123456': 6}`

C `{'123': 3, '12345': 5, '1': 1}`

D `{'12345678': 8, '1': 1, '1234': 4, '12': 2, '6': 1, '123456': 6}`

Question 18 – Function–Loop–2 – 355296.3.11

Suppose you have the following function `count_characters`, which takes two arguments:
- a list of strings which are names, and
- an optional keyword argument called `character` that has a default value of `'a'`.

```
def count_character(x, character='a'):
    y = {}
    for key in x:
        count = 0
        for char in key:
            if char == character:
                count += 1
        y[key] = count
    return y
```

What would be returned by this function, if we call the function twice as follows:
```
count_character(['Emma', 'Julia', 'Ian', 'Noah', 'Liam', 'Anna'], 'i')
count_character(['Emma', 'Julia', 'Ian', 'Noah', 'Liam', 'Anna'])
```

A
```
{'Emma': 0, 'Julia': 1, 'Ian': 0, 'Noah': 0, 'Liam': 1, 'Anna': 0}
{'Emma': 1, 'Julia': 1, 'Ian': 1, 'Noah': 1, 'Liam': 1, 'Anna': 1}
```

B
```
{'Emma': 0, 'Julia': 1, 'Ian': 0, 'Noah': 0, 'Liam': 1, 'Anna': 0}
{'Emma': 0, 'Julia': 0, 'Ian': 0, 'Noah': 0, 'Liam': 0, 'Anna': 0}
```

C The first function call will result in an error because there are too many arguments.

D The second function call will result in an error because there is a missing argument.

Question 19 – lists – lambda – 1.1 – 354850.1.3

You have four different blocks of code. Three of them return the same list, one of them returns something different.

Which block of code prints something different than the other blocks print?

A
```
print(list(map(lambda x: x**2, [1, 2, 3])))
```

B
```
def squared_function(x):
    return x**2
print(list(map(squared_function, [1, 2, 3])))
```

C
```
def squared_function(x):
    return x**2
print(squared_function([1, 2, 3]))
```

D
```
squared_numbers = lambda x: x**2
print([squared_numbers(1), squared_numbers(2), squared_numbers(3)])
```

Question 20 − lists − loop − 1.1 − 354625.1.4

What is the output of the program below?

```
x = [1, 2, 3, 4, 5, 6, 7, 8]
count = [None] * len(x)

for i in range(len(x)):
    if x[i] > 4:
        count[i] = True
    else:
        count[i] = False

sum(count)
```

A 4

B 0

C `None`

D This code will result in an error as you cannot multiple `None` with an integer.

Question 21 − lists − loop − 1.2 − 354626.1.5

Assume you already have a variable called `x`, which contains a list of integers.

Which of the following programs will print you a list of the squared values of the elements in `x`?

For example:
If `x = [1, 2, 4, 8]`
the program should prints:
`[1, 4, 16, 64]`.

A
```
mylist = [0] * len(x)
for i in range(len(x)):
    mylist[i] = x[i]**2
print(mylist)
```

B
```
mylist = [0] * len(x)
for i in x:
    mylist[i] = x[i]**2
print(mylist)
```

C
```
mylist = []
for i in range(len(x)):
    mylist[i] = x[i]**2
print(mylist)
```

D
```
mylist = []
for i in x:
    mylist[i] = x[i]**2
print(mylist)
```

Question 22 – lists – straightforward – 1.1 – 354600.1.3

Consider the following code snippet:

```
x = ["a", "b", "c", "4"]
print(x[-3] + x[3]*3)
```

Which of the following code snippet will print the same output?

A
```
z = [["a", "b", "c"], [1, 2, 3, 4]]
print(z[0][1] + str(z[1][3])*3)
```

B
```
print('b'+ str(8 % 4)*3)
```

C
```
print('b' + str(len(x)*3))
```

D
```
y = [1, 2, 3, 4]
print('b'+ max(y)*3)
```

Question 23 – lists – straightforward – 3 – 354634.1.6

Consider the following code snippet:

```
mylist = [1, 2, 5, 7, 9, 12]
new_list1 = mylist[0:3]
new_list2 = mylist[1:4]
print(new_list1 + new_list2)
```

What will be printed?

A `[1, 2, 5, 2, 5, 7]`

B `[1, 2, 5, 7, 2, 5, 7, 9]`

C `[2, 5, 7, 5, 7, 9]`

D `[2, 5, 5, 7]`

Question 24 – lists – straightforward – 4 – 354636.1.4

You have the following list:

```
my_list = [1,2,3,4]
```

Which one of the following lines of code lines does **not** print `4` as the output?

A
```
print(max(my_list))
```

B
```
print(len(my_list))
```

C
```
print(my_list[4])
```

D
```
print(my_list[-1])
```