Test Trial midterm - Introduction to Python - 22/23

UNIVERSITEIT VAN AMSTERDAM

Question 1 − Basics−Logic−3 − 355237.1.2

What is the output of the following code snippet?

```
x = 1
y = 15
print(not(not(x < 3) and not(y > 14 or y > 10)))
```

[redacted]

The code will result in a `SyntaxError`.

`True`

`False`

`0`

Question 2 − Basics−Loop−2 − 354680.1.0

What is the output of the following code snippet?

```
list1 = [1, 2, 3, 4, 5]
list2 = [2, 4, 6, 8, 10]
list3 = []

for i in list1:
    for j in list2:
        if i + j >= 10:
            break
        list3.append(i + j)

print(list3)
```

[redacted]

`[3, 5, 7, 9, 4, 6, 8, 5, 7, 9, 6, 8, 7, 9]`

`[3, 5, 7, 8, 9, 6, 8, 10]`

`[3, 5, 7, 8, 9, 6, 8, 9]`

This code will result in an `IndexError`.

Question 3 − Basics−Scope−2 − 354721.2.0

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

```
def outer():
    a = 5
    def inner():
        a = 10
        print("Value of a inside inner():", a)
    inner()
    print("Value of a inside outer():", a)
outer()
```

[redacted]

```
x = 10
def outer():
    x = 5
    def inner():
        global x
    inner()
    print("Value of x inside outer():", x)
outer()
print("Value of x outside outer():", x)
```

```
x = 10
def outer():
    x = 5
    def inner():
        nonlocal x
    inner()
    print("Value of x inside outer():", x)
outer()
print("Value of x outside outer():", x)
```

```
def outer():
    x = 10
    def inner():
        global x
        x = 5
    inner()
    print("Value of x inside outer():", x)
outer()
print("Value of x outside outer():", x)
```

```
def outer():
    x = 5
    def inner():
        nonlocal x
        x = 10
    inner()
    print("Value of x inside outer():", x)
outer()
print("Value of x outside outer():", x)
```

Question 4 − Basics−Variables−1 − 354629.1.3

Assume you have the following variables:

```
x = 'abc'
y = 3
z = 2.0
```

What will be printed after these lines of code?

```
print(x*y)
print(x*z)
print(y*z)
```

[redacted]

`'abcabcabc'`
The code will result in a `TypeError` because you can't multiply a float and a string together.
`6.0`

The code will result in a `ValueError` because you can't concatenate a string and an integer.
The code will result in a `TypeError` because you can't multiply a float and a string together.
`6`

`'abcabcabc'`
`'abcabc'`
`6.0`

`'abcabcabc'`
The code will result in a `TypeError` because you can't multiply a float and a string together.
`6`

Question 5 − dictionaries − logic − 1.1 − 354624.1.1

What is the output of the program below?

```
x = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
y = {1: 50, 2: 40, 3: 30, 4: 20, 5: 10}
z = x.copy()

for key, value in x.items():
    if key in y:
        if value <= y[key]:
            z[key] = y[key]
    z[key] = y[key]

print(z)
```

[redacted]

`{1: 50, 2: 40, 3: 30, 4: 20, 5: 10}`

`{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}`

None of the given options

`{}`

Question 6 − dictionaries − loop − 1 − 354603.1.3

Suppose you have the following dictionary:

```
sample_dict = {
    "name": "Kim",
    "age": 25,
    "birthdate": "3-2-1998",
    "city": "Amsterdam"}
```

Suppose you want to create a dictionary that only contains the keys 'name' and 'city'.

Which of the following code snippets return(s) the desired dictionary?

[redacted]

```
keys = ["name", "city"]
new_dict = {}
for k in keys:
    new_dict[k] = sample_dict[k]
new_dict
```

```
keys = ["age", "birthdate"]
new_dict = {}
new_keys = sample_dict.keys() - keys
for k in new_keys:
    new_dict[k] = sample_dict[k]
new_dict
```

None of the given options

Both of the given options

Question 7 − dictionaries − straightforward − 1.1 − 354897.1.3

Suppose you have the following dictionary:

```
closet = {
    "shirts" : 5,
    "colors" : ['red', 'yellow', 'blue', 'pink']
}
```

You want to:
- add a key "`shoes`" which has a value that is a list containing the strings "`sneakers`" and "`heels`"
- add 2 shirts, so that the value of the key "`shirts`" becomes 7

So the final dictionary should look like:

```
{'shirts': 7, 'colors': ['red', 'yellow', 'blue', 'pink'], 'shoes': ['sneakers', 'heels']}
```

Which of the programs below will work as intended?

[redacted]

```
shoes = {'shoes': ['sneakers', 'heels']}

closet["shirts"] = 7
closet = dict(zip(closet, shoes))
```

```
closet.keys() = closet.keys() + "shoes"

closet.values() = [7, ['red', 'yellow', 'blue', 'pink'], ['sneakers', 'heels']]
```

Both of the given options

None of the given options

Question 8 − dictionaries − straightforward − 2.2 − 355318.1.2

Suppose you have the following function:

```
def my_count_function(string):
    counts = {}
    for letter in string:
        counts[letter] = string.count(letter)
    return counts
```

And you create the following two dictionaries:

```
desk_count = my_count_function("desk")
laptop_count = my_count_function("laptop")
```

Which of the lines of code below returns something different than the rest?

[redacted]

```
len(desk_count)

max(laptop_count.values())*2

sum(desk_count.values())

sum(laptop_count.keys())
```

Question 9 − Function−Argument−Default−1 − 355271.2.1

Suppose you have the following function:

```
def add_numbers(num1, num2 = 10, num3 = 20):
    return num1 + num2 + num3
```

What are the outputs if we call the function three times as follows:

```
add_numbers(5, 15)
add_numbers(5, num3 = 30)
add_numbers(num2 = 15, num3 = 10)
```

[redacted]

```
40
45
25
```

```
40
45
```
The code will result in `TypeError` because there is a missing argument.

```
25
35
35
```

All three lines of code will result in `TypeError` because there is a missing argument.

Question 10 − Function−Argument−Flexible−Keyword−1 − 355285.1.1

You need to write a function called `print_info` which takes two required arguments called `name` and `age`, and a flexible number of keyword arguments.

The function should return a dictionary with `name`, `age`, and all other key-value pairs that may be passed as keyword arguments.

For example:

If we call the function as:
`print_info('John', 30, city = 'New York', job = 'software engineer')`

then it should return the dictionary:
`{'name' = 'John', 'age' = 30, 'city' = 'New York', 'job' = 'software engineer'}`

Which of the following code snippets will do what you want?

[redacted]

```
def print_info(name, age, **kwargs):
    info = {}
    info['name'] = name
    info['age'] = age
    for key, value in kwargs.items():
        info[key] = value
    return info
```

```
def print_info(**kwargs, name, age):
    for key, value in kwargs:
        info = {}
        info[key] = value
        info['name'] = name
        info['age'] = age
    return info
```

```
def print_info(name, age, **kwargs):
    info = {}
    info.append(name)
    info.append(age)
    for element in kwargs:
        info.append(element)
    print(info)
```

```
def print_info(name, age, **kwargs):
    info = {}
    info['name'] = name
    info['age'] = age
    for key in kwargs.keys() and value in kwargs.values():
        info[key] = value
    return info
```

Question 11 − Function−Lambda−1 − 355288.1.5

You need to write a function called `sort_list` that accepts a list of integers.

The function should return a new list, in which the elements are sorted from highest to lowest based on their absolute values.

For example:
If we call the function as:
`sort_list([-9, 2, 5, -3, -10, 4, 7])`

then it should return the list:
`[-10, -9, 7, 5, 4, -3, 2]`

Which of the following functions will give you the correct output?

[redacted]

```
def sort_list(x):
    return sorted(x, key = lambda num: abs(num), reverse = True)
```

```
def sort_list(x):
    return sorted(nums, key = lambda x: abs(x), reverse = True)
```

```
def sort_list(x):
    return x.sort(key = lambda num: abs(num), reverse = True)
```

All of three functions will give the correct output.

Question 12 − Function−Logic−1 − 354637.1.7

You want to write a program to calculate your income tax for 2023.

Suppose your country imposes the following tax brackets:

Bracket	Tax rate	Taxable income	
		Over	Not over
1	10%	$0	$11,000
2	12%	$11,000	$45,000
3	22%	$45,000	$95,000
4	32%	$95,000	

If your gross income is $50,000, you will pay 10% tax on the first $11,000, 12% tax on the next $34,000 and 22% tax on the last $5,000.

Which of the following function will return the output containing your `net_income` and your `total_tax` given your `gross_income`?

For example:
If we call the function as:
`income_tax_calculator(50000)`
it should return:
`{'total_tax': 6280.0, 'net_income': 43720.0}`

```
def income_tax_calculator(gross_income):

    results = {}

    rate_1 = 0.10
    rate_2 = 0.12
    rate_3 = 0.22
    rate_4 = 0.32

    bracket_1 = 11000
    bracket_2 = 45000 - 11000
    bracket_3 = 95000 - 45000

    if gross_income > 95000:
        total_tax = ((gross_income - 95000) * rate_4) + (bracket_3 * rate_3) + (bracket_2 * rate_2)
+ (bracket_1 * rate_1)
    elif 95000 >= gross_income > 45000:
        total_tax = ((gross_income - 45000) * rate_3) + (bracket_2 * rate_2) + (bracket_1 * rate_1)
    elif 45000 >= gross_income > 11000:
        total_tax = ((gross_income - 11000) * rate_2) + (bracket_1 * rate_1)
    else:
        total_tax = gross_income * rate_1

    net_income = gross_income - total_tax

    results['total_tax'] = round(total_tax, 2)
    results['net_income'] = round(net_income, 2)

    return results

def income_tax_calculator(gross_income):

    results = {}

    rate_1 = 0.10
    rate_2 = 0.12
    rate_3 = 0.22
    rate_4 = 0.32

    bracket_1 = 11000
    bracket_2 = 45000 - 11001
    bracket_3 = 95000 - 45001

    total_tax = (gross_income - 95001) * rate_4 + (bracket_3 * rate_3) + (bracket_2 * rate_2) +
(bracket_1 * rate_1)
    net_income = lambda gross_income, total_tax: gross_income - total_tax

    results['total_tax'] = round(total_tax, 2)
    results['net_income'] = round(net_income, 2)

    return results
```

```
def income_tax_calculator(gross_income):

    results = {}

    rate_1 = 0.10
    rate_2 = 0.12
    rate_3 = 0.22
    rate_4 = 0.32

    if gross_income <= 11000:
        total_tax = gross_income * rate_1
    if 45000 > gross_income > 11000:
        total_tax += (45000 - gross_income) * rate_2
    if 95000 > gross_income > 45000:
        total_tax += (95000 - gross_income) * rate_3
    else:
        total_tax += (gross_income - 95000) * rate_4

    net_income = gross_income - total_tax

    results['total_tax'] = total_tax
    results['net_income'] = net_income

    return(total_tax, net_income)
```

All of three functions will work as intended.

Question 13 − lists − lambda − 1.2 − 355328.1.4

Suppose you have the following functions, one to elevates a number to square and the second elevates to cube:

```
def square(n):
    return (n**2)
def cube(n):
    return (n**3)
```

You want to apply both functions to the elements in a list at the same time.

For example:

If you have a list such as:

```
my_list = [0, 1, 2, 3, 4]
```

you want to print:

```
[0, 0]
[1, 1]
[4, 8]
[9, 27]
[16, 64]
```

Which of the following blocks of codes works as intended?

[redacted]

```
funcs = [square, cube]

for i in my_list:
    results = map(lambda x: x(i), funcs)
    print(list(results))
```

```
funcs = [square, cube]

for i in my_list:
    results = map(lambda x: x[i], funcs)
    print(list(results))
```

```
funcs = [square, cube]

for i in my_list:
    results = map(lambda x: x(i), funcs)
    print(results)
```

```
funcs = [square, cube]

for i in my_list:
    results = map(lambda x: x[i], funcs)
    print(results)
```

Question 14 – lists – loop & logic (enumerate) – 1 – 354855.3.0

Assume you already have a variable called alphabet, which contains a string, e.g., `alphabet = "abcdefghijklmnopqrstuvwxyz"`

Which of the following code snippets will give the same output as the following command?
`print(list(alphabet[1::2]))`

Hint: The `reverse()` method reverses the sorting order of the elements in a list.

[redacted]

```
values = []
for index, value in enumerate(alphabet, start=1):
    if index % 2 == 0:
        values.append(value)
print(values)
```

```
values = []
for index, value in enumerate(alphabet[::-1]):
    if index % 2 == 0:
        values.append(value)
        values.reverse()
print(values)
```

Both of the given options.

None of the given options.

Question 15 – lists – straightforward – 1.2 – 354832.1.2

Assume you already have a variable `a`, which is an integer between 1 and 4.

What is the output of the following code snippet?

```
mylist = [1, 3, 5, 7, 9]
print(mylist[a] + mylist[-a])
```

[redacted]

`12`

`11`

`10`

`8`

Question 16 − lists − straightforward − 2 − 354602.1.6

Consider the following code snippet:

```
mylist = [1, 3, 2, 3, 4, 5, 3, 3]
mylist.append('3')
mylist.remove(3)
print(mylist.count(3))
```

What will be the output?

Hint: The `remove()` list method removes the first occurrence of the element with the specified value.

[redacted]

3

4

0

None of the given options.