Python for Data Science - Mock Exam

Table of contents

Part 1

Assignment 1: Basics

Task 1.1

The following script is given:

```
var1 = (42, 0.34)
var2 = 0.34
var3 = {elem:1 for elem in var1}
var4 = set(list(range(3)))
```

- What data type is var1: `tuple`
- What data type is var2: `float`
- What data type is var3: `dict`
- What data type is var4: `set`

Prove

```
var1 = (42, 0.34)
print(type(var1))

var2 = 0.34
print(type(var2))

var3 = {elem:1 for elem in var1}
print(type(var3))

var4 = set(list(range(3)))
print(type(var4))
```

```
<class 'tuple'>
<class 'float'>
<class 'dict'>
<class 'set'>
```

Task 1.2

What does the script below print to the console?

```
a = 14
b = "this variable"
c = "formatting"
mylist = [a,b,c]
mystring = "Hello, {} used {} syntax '.format'." .format(mylist[1], mylist[2])
print(mystring)
```

Output:

```
Hello, this variable used formatting syntax '.format'.
```

Prove

```
a = 14
b = "this variable"
c = "formatting"
mylist = [a,b,c]
mystring = "Hello, {} used {} syntax '.format'." .format(mylist[1], mylist[2])
print(mystring)
```

```
Hello, this variable used formatting syntax '.format'.
```

Task 1.3

Given the following script:

```
a = "a"
mystring = "I'm just {} normal sentence {}".format(__FILL_ONE__, __FILL_TWO__)
print(mystring)
```

Please provide suitable replacements for `__FILL_ONE__` and `__FILL_TWO__` such that the script prints: I'm just a normal sentence !

- `__FILL_ONE__`: a
- `__FILL_TWO__`: "!"

Prove

```
a = "a"
mystring = "I'm just {} normal sentence {}".format(a, "!")
print(mystring)
```

```
I'm just a normal sentence !
```

Assignment 2: Functions

Task 2.1

What does the script below print to the console?

```
a = 2
b = 5

def mathematical_fun(a, b):
    return a+b

print(a, mathematical_fun(a, b))
```

Output:

```
2 7
```

Prove

```
a = 2
b = 5

def mathematical_fun(a, b):
    return a+b

print(a, mathematical_fun(a, b))
```

2 7

Task 2.2

What does the script below print to the console?

```
a = 3

def mathematical_fun(a, b=2):
    return a+b

result = mathematical_fun(a)
print(result)
```

Output:

```
5
```

Prove

```
a = 3

def mathematical_fun(a, b=2):
    return a+b

result = mathematical_fun(a)
print(result)
```

5

Task 2.3

What does the script below print to the console?

```
var = 2

def change_value(var):
    print(var)
    var = 3
```

```
    return var

print(var)
```

Output:

```
2
```

Prove

```
var = 2

def change_value(var):
    print(var)
    var = 3
    return var

print(var)
```

2

Task 2.4

What does the script below print to the console?

```
MODULE_VAR = 2

def module_function(a):
    global MODULE_VAR
    print(MODULE_VAR)
    MODULE_VAR = a

module_function(3)
print(MODULE_VAR)
```

Output:

```
2
3
```

Prove

```
MODULE_VAR = 2

def module_function(a):
    global MODULE_VAR
    print(MODULE_VAR)
    MODULE_VAR = a

module_function(3)
print(MODULE_VAR)
```

```
2
3
```

Assignment 3: Control Structures

Task 3.1

What does the script below print to the console?

```
for elem in range(2):
    print(elem)
```

Output:

```
0
1
```

Prove

```
for elem in range(2):
    print(elem)
```

```
0
1
```

Task 3.2

What does the script below print to the console?

```
mylist = [1,2,3]
for ind, elem in enumerate(mylist):
    print(ind)
print(elem)
```

Output:

```
0
1
2
3
```

Prove

```
mylist = [1,2,3]
for ind, elem in enumerate(mylist):
    print(ind)
print(elem)
```

```
0
1
2
3
```

Task 3.3

What does the script below print to the console?

```
a = 21
b = 2
if len("155kl") > 4:
    while a < 24:
        print(a)
        if b <= 2:
            break
        a += 1
```

```
else:
    while a < 23:
        print(a + 1)
        if b <= 2:
            break
        a += 1
```

Output:

```
21
```

Prove

```
a = 21
b = 2
if len("155kl") > 4:
    while a < 24:
        print(a)
        if b <= 2:
            break
        a += 1
else:
    while a < 23:
        print(a + 1)
        if b <= 2:
            break
        a += 1
```

21

Assignment 4: Complex Scripts, OOP

Task 4.1

What does the script below print to the console?

```
class MyClass ():
    number = 3

    def __init__(self, number):
```

```
        self.number = number

obj = MyClass(4)
print(MyClass.number)
```

Output:

```
3
```

Prove

```
class MyClass ():
    number = 3

    def __init__(self, number):
        self.number = number

obj = MyClass(4)
print(MyClass.number)
```

3

Task 4.2

What does the script below print to the console?

```
class MyNumber():
    def __init__(self, number):
        self.number = number

    def my_method(self):
        self.number = self.number + 1
        return self.number

i = 0
obj = MyNumber(i)
while i <= 3:
    i = obj.my_method()
    print(i)
else:
    print("?")
```

Output:

```
1
2
3
4
?
```

Prove

```
class MyNumber():
    def __init__(self, number):
        self.number = number

    def my_method(self):
        self.number = self.number + 1
        return self.number

i = 0
obj = MyNumber(i)
while i <= 3:
    i = obj.my_method()
    print(i)
else:
    print("?")
```

```
1
2
3
4
?
```

Task 4.3

What does the script below print to the console?

```
class MyBaseClass():
    def __init__(self, number):
        self.number = number
```

```
    def my_method(self):
        self.number = self.number + 10
        return self.number

class MyClass(MyBaseClass):
    def __init__(self, number):
        super().__init__(number)

    def my_method(self):
        self.number = self.number + 1
        return self.number

obj = MyClass(3)
print(obj.number)
print(obj.my_method())
print(obj.number)
```

Output:

```
3
4
4
```

Prove

```
class MyBaseClass():
    def __init__(self, number):
        self.number = number

    def my_method(self):
        self.number = self.number + 10
        return self.number

class MyClass(MyBaseClass):
    def __init__(self, number):
        super().__init__(number)

    def my_method(self):
        self.number = self.number + 1
        return self.number
```

```
obj = MyClass(3)
print(obj.number)
print(obj.my_method())
print(obj.number)
```

```
3
4
4
```

Assignment 5: Multiple Choice

Task 5.1

Which of the following statements about Python dictionaries are correct?

> Note: At least one statement must be marked, otherwise you will get 0 points.

- [ ] You can modify the value associated with a key in a dictionary after it has been created.
- [o] Keys in a dictionary must be strings.
- [o] Values in a dictionary must be of the same data type.
- [o] A dictionary can contain duplicate keys.
- [ ] Dictionaries can form nested structures.

Task 5.2

Which of the following statements about Python classes/objects are correct?

> Note: At least one statement must be marked, otherwise you will get 0 points.

- [ ] A class is a blueprint for creating objects.
- [ ] An object can have attributes and methods.
- [ ] A class can be instantiated multiple times to create different objects.
- [o] All objects in Python are instances of built-in classes only.
- [ ] Inheritance allows a class to inherit attributes and methods from another class.

Part 2

Assignment 1: Quick Questions

Question 1

- Q: Is the data_processor.py module depending on any other custom modules?
- A: No, there is no explicit dependency on any other custom modules.

Question 2

- Q: How many and which classes inherit from DataProcessor class?
- A: There is one class that inherits from the `DataProcessor` class in the provided module: `DataAnalyzer`.

Question 3

- Q: Would the methods from DataLoader class still work if the first import statement (first line) of the module data_loader.py was missing? Justify your answer in one statement.
- A: No, the `load_data` method would fail because it explicitly relies on the `csv` module.

Assignment 2: Code Completions

ToDo 1

Import the classes DataProcessor and DataAnalyzer.

```
from data_processor import DataProcessor, DataAnalyzer
```

ToDo 2

Call the method load_data from loader object.

```
loader.load_data()
```

ToDo 3

What is the data type of the resulting data variable?

```
dict
```

ToDo 4

Is it possible to print an object of type ExcelDataLoader? Justify your answer with one statement.

Short Answer: Yes, the `__str__` method (and also `__repr__`) is defined by default for every class in Python, because all classes implicitly inherit from the base class object.

Long Answer and Justification: If you do not define your own`__str__` method in a class like ExcelDataLoader, the implementation from the base class object is used. However, this default implementation is very general and usually only provides a technical description of the object, such as:

```
<__main__.ExcelDataLoader object at 0x105a30e80>
```

This output is rarely helpful to the user, as it provides no meaningful information about the content or state of the object (e.g., the loaded data or the file_path).

ToDo 5

How do you print the object analyzer and what output do you expect (sketch it down)?

```
print(analyzer)
```

Output:

```
sepal_length        sepal_width         petal_length        petal_width         species
4.4                 3.0                 1.3                 0.2                 setosa
7.2                 3.2                 6.0                 1.8                 virginica
6.4                 3.1                 5.5                 1.8                 virginica
4.9                 NA                  1.4                 0.2                 setosa
5.7                 3.8                 1.7                 0.3                 setosa
```

ToDo 6

Write a function double_fun, that doubles a value (parameter input) and returns it.

```
def double_func(value):
    return value*2
```

ToDo 7

Apply the function double_fun together with the analyzer object to double all values of petal_length.

```
analyzer.apply_function("petal_length", double_func)
```

ToDo 8

Count the number of versicolor entries in the list species: 4 Points

> **Note**
>
> Only one of the following alternatives is evaluated: * If you solve TODO 8 using list comprehension -> (4 points) * If you solve TODO 8 without list comprehension -> (3 points) * If you have problems with the previous definitions or dependencies, solve the independent task: TODO 8 B: Count the number of zeros in zeros -> (2 points)

```
res = sum(1 for i in species if i == "versicolor")
```