Python for Data Science - Modul Exam FS2023

Table of contents

Task 1: Slicing

For each of the following code snippets: What will be printed?

Task 1.1

```
s = "Mojo is the successor programming language to Python!"
print(s[12:19])
```

Output:

```
success
```

Prove

```
s = "Mojo is the successor programming language to Python!"
print(s[12:19])
```

```
success
```

Task 1.2

```
s = "Mojo is the successor programming language to Python!"
print(s[12::4])
```

Output:

```
seromgngot!
```

Prove

```
s = "Mojo is the successor programming language to Python!"
print(s[12::4])
```

```
seromgngot!
```

Task 1.3

```
s = "Mojo is a language"
print(s[::-1])
```

Output:

```
egaugnal a si ojoM
```

Prove

```
s = "Mojo is a language"
print(s[::-1])
```

```
egaugnal a si ojoM
```

Task 1.4

```
L = ["Mojo", "is", "a", "language"]
print(L[3][:-4])
```

Output:

```
lang
```

Prove

```
L = ["Mojo", "is", "a", "language"]
print(L[3][:-4])
```

lang

Task 2: Pack/Unpacking, Lambda

The following list and the two functions are given:

```
letter_list = ["h", "s", "l", "u", " ", "m", "e", "p", "!"]

def upper_with_for_loop(letter_list):
    squares = []
    for x in letter_list:
        squares.append(x.upper())
    return squares

def upper_with_lambda(letter_list):
    return list(map(lambda x: x.upper(), letter_list))
```

For each of the following code snippets: What will be printed?

Task 2.1

```
print("1) for_loop_a:", upper_with_for_loop(letter_list))
print("2) for_loop_b:", *upper_with_for_loop(letter_list), sep="")
```

Output:

```
1) for_loop_a: ['H', 'S', 'L', 'U', ' ', 'M', 'E', 'P', '!']
2) for_loop_b:HSLU MEP!
```

Prove

```
print("1) for_loop_a:", upper_with_for_loop(letter_list))
print("2) for_loop_b:", *upper_with_for_loop(letter_list), sep="")
```

```
1) for_loop_a: ['H', 'S', 'L', 'U', ' ', 'M', 'E', 'P', '!']
2) for_loop_b:HSLU MEP!
```

Task 2.2

```
print("3) lambda_a:", upper_with_lambda(letter_list))
print("4) lambda_b:", *upper_with_lambda(letter_list))
```

Output:

```
3) lambda_a: ['H', 'S', 'L', 'U', ' ', 'M', 'E', 'P', '!']
4) lambda_b: H S L U   M E P !
```

Prove

```
print("3) lambda_a:", upper_with_lambda(letter_list))
print("4) lambda_b:", *upper_with_lambda(letter_list))
```

```
3) lambda_a: ['H', 'S', 'L', 'U', ' ', 'M', 'E', 'P', '!']
4) lambda_b: H S L U   M E P !
```

Task 3: OO Inheritance

For each of the following code snippets: What will be printed?

Task 3.1

```
class AAA:
    def m(self):
        print("AAA")

class BB(AAA):
    def m(self):
        print("BB")

class CC(AAA):
    def m(self):
        print("CC")

class D(BB, CC):
    def m(self):
        print("D")
```

```
d = D()
d.m()
```

Output:

```
D
```

Prove

```
d = D()
d.m()
```

D

Task 3.2

```
class AAA:
    def m(self):
        print("AAA")

class BB(AAA):
    def m(self):
        print("BB")

class CC(AAA):
    def m(self):
        print("CC")

class D(BB, CC):
    pass
```

```
d = D()
d.m()
```

Output:

```
BB
```

Prove

```
d = D()
d.m()
```

```
BB
```

Task 3.3

```
class AAA:
    def m(self):
        print("AAA")
```

```
class BB(AAA):
    pass

class CC(AAA):
    def m(self):
        print("CC")

class D(BB, CC):
    pass
```

```
d = D()
d.m()
```

Output:

```
CC
```

Prove

```
d = D()
d.m()
```

CC

Task 3.4

```
class AAA:
    def m(self):
        print("AAA")

class BB(AAA):
    pass

class CC(AAA):
    pass

class D(BB, CC):
    pass
```

```
d = D()
d.m()
```

Output:

```
AAA
```

Prove:

```
d = D()
d.m()
```

```
AAA
```

Task 4 String: C-style / format()

For two following code snippets (Task 4.1 and Task 4.2): What will be printed?

> **Caution**
>
> Your inputs are given in the comments!

Task 4.1

```
name = input("Enter your name: ") # your input: Anna
age = int(input("Enter your age: ")) # your input: 23

print("My name is %s and I am %d years old." % (name, age))
```

Output:

```
My name is Anna and I am 23 years old.
```

Prove

```
name = "Anna"
age = 23

print("My name is %s and I am %d years old." % (name, age))
```

```
My name is Anna and I am 23 years old.
```

Task 4.2

```
first_name = input("Enter your first name: ") # your input: Anna
last_name = input("Enter your last name: ") # your input: Meier
birth_year = input("Enter your birth year: ") # your input: 1998
username = "%s%s%s" % (first_name[0].lower(), last_name[:3].lower(), birth_year[-2:])

print("Generated username: %s" % username)
```

Output:

```
Generated username: amei98
```

Prove

```
first_name = "Anna"
last_name = "Meier"
birth_year = "1998"
username = "%s%s%s" % (first_name[0].lower(), last_name[:3].lower(), birth_year[-2:])

print("Generated username: %s" % username)
```

```
Generated username: amei98
```

Task 4.3

For the following code snippet: Change the C-style string format below to the “pythonic way” with the format()-function. Write the “pythonic way” with the `format()`-function as result.

> **Caution**
>
> Your inputs are given in the comments!

```
name = input("Enter your name: ") # your input: Anna
age = int(input("Enter your age: ")) # your input: 23

# replace the next line with a 'format(...)' function:
print("Hello, my name is %s and I am %d years old." % (name, age))
```

Solution:

```
print("Hello, my name is {} and I am {} years old.".format(name, age))
```

Output:

```
My name is Anna and I am 23 years old.
```

Prove

```
name = "Anna"
age = 23

print("Hello, my name is {} and I am {} years old.".format(name, age))
```

```
Hello, my name is Anna and I am 23 years old.
```

Task 4.4

For the following code snippet: Change the C-style string format below to the "pythonic way" with the format()-function. Write the "pythonic way" with the `format()`-function as result.

> **Caution**
>
> Your inputs are given in the comments!

```
first_name = input("Enter your first name: ") # your input: Anna
last_name = input("Enter your last name: ") # your input: Meier
birth_year = input("Enter your birth year: ") # your input: 1998

# replace the next two code lines with 'format(...)' functions:
username = "%s%s%s" % (first_name[0].lower(), last_name[:3].lower(), birth_year[-2:])
print("Generated username: %s" % username)
```

Solution:

```
print("Generated username: {}".format(username))
```

Output:

```
Generated username: amei98
```

Prove

```
first_name = "Anna"
last_name = "Meier"
birth_year = "1998"
username = "%s%s%s" % (first_name[0].lower(), last_name[:3].lower(), birth_year[-2:])

print("Generated username: {}".format(username))
```

```
Generated username: amei98
```

Task 5: True/False Statements

Which of the following statements are true? Select only the correct statements!

> **Caution**
>
> Each false selection of a statement gives a 1/2 point deduction.

- [ ] Examples of non-Iterable are Integer, Float.
- [ ] Generator, Files and Range are all iterables.
- [ ] Integer and floats are non-sequence types.

- [ ] An iterator is always an iterable.
- [ ] Iterators always must have a `__iter__()` and `__next__()` method.
- [o] A generator expression is never an iterable.
- [o] A generator function is an iterable.
- [o] Byte and Range are not iterable.