Modul exam PDS

FS2023 **15.6.2023**

Specification of the candidates:	
Name and first name:	

Information about the exam:			
Duration of the exam:	25 min		
Scoring:	**Tasks**	**max. points**	**Points scored**
	1: Slicing	4	
	2: Pack/Unpacking/Lambda	4	
	3: OO Inheritance	4	
	4: String: C-style / format()	4	
	5: True/False Statements	4	
	Total	**20**	
	Note		
Tools:	▪ OpenBook (Paperware only!!!) ▪ no mobile phones, smartwatches, laptops, tablets, calculators, etc.		
Lecturers:	Ramón Christen, Prof. Erwin Mathis, Nicola Brandenberg		

Task 1 Slicing 4 pts

For each of the following code snippets:

- What will be printed?
- Write the result in the empty text field below.

Task 1.1

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

Task 1.2

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

Task 1.3

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

Task 1.4

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

Task 2 Pack/Unpacking, Lambda 4 pts

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?
- Write the result in the empty text field below.
- Every space has to be marked with a sign like: ␣

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="")
```

Task 2.2

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

Task 3 OO Inheritance 4 pts

For each of the following code snippets:

- What will be printed?
- Write the result in the empty text field below.

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()
```

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()
```

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()
```

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()
```

Task 4 String: C-style / format() 4 pts

For two following code snippets (Task 4.1 and Task 4.2):

- What will be printed?
- Write the result of the **print()** Statements in the empty text field below.

Attention: 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))
```

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)
```

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 in the empty text field below.

Attention: 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))
```

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 in the empty text field below.

Attention: 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)
```

Task 5 True/False Statements 4 pts

Which of the following statements are true? Select only the correct statements! ☒ = true
Attention: 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.
- ☐ A generator expression is never an iterable.
- ☐ A generator function is an iterable.
- ☐ Byte and Range are not iterable.