

Programming

Winter 2020/2021

Number 03, Submission Deadline: Dec.01, 2020

1. Use a for-loop to compute the *arithmetic mean* of the following list of (2 P)
numbers:

```
[87, 98, 95, 9, 80, 70, 1, 43, 92, 23]
```

2. Python provides a module called random for generating pseudo-randomized (2 P)
numbers. Use the random() function of this module to sample pseudo-
random floating point numbers from the interval [0,1). Use a while-
loop to count the number of samples needed to receive a pseudo-random
number that is smaller than a given threshold value a , e.g., say $a = 0.1$.

3. Write a function getDuplicates(lst, x) that returns elements that oc- (2 P)
cur at least x times in the given list lst. Do not use the list's count()
function in your implementation. Make sure to report each such identi-
fied duplicate element only once. You may assume that all elements of
list lst are immutable. Test your function on the following list, using
different values for threshold x.

```
[2, 'test ', 2, (1, 2), 3, 2, 'test ', (1, 2), 1, 2, 4, 3]
```

4. Functions can have optional parameters by assigning default values. (2 P)
Note that optional parameters must always succeed required paramet-
ers in order, as illustrated in this example:

```
def myFunction(a, b=None, c=1, d='text '):  
    return (a, b, c, d)
```

```
myFunction(42)  
# will return (42, None, 1, 'text ')  
myFunction(list(), set())  
# will return ([], set(), 1, 'text ')  
myFunction(list(), d='Hello World', c=3)  
# will return ([], None, 3, 'Hello World')
```

Extend the *constructor*, i.e., the `__init__()` function, of the Library
class shown in the lecture by default parameters so that (i) it can be
called without any required arguments and (ii) an initial list of books
can be supplied.

5. Explain in your own words the difference between class and instance variables/functions. Create a class of your own to illustrate your explanation. (3 P)
6. The following function performs a matrix multiplication of the given two-dimensional matrices M1 and M2. Identify the assumptions that the code makes and translate these into Python assert statements that are checked prior to the code that performs the matrix multiplication. Augment each assertion with a meaningful message (as shown in the lecture) so that the user is properly informed in case the assertion is not met. (3 P)

```
def matrixMultiplication(M1, M2):
    # insert assertions here ...
    m = len(M1)
    n = len(M2[0])

    M3 = list()
    for i in range(m):
        M3.append(list())
        for j in range(n):
            M3[-1].append(0)
            for k in range(len(M1[i])):
                M3[i][j] += M1[i][k] * M2[k][j]
    return M3
```

Important:

Please submit your solution as (adequately commented) Python file. Use the cell separator comment “`#%%`” to partition your Python file analog to the six exercises. Make sure your Python file contains only valid Python code.