Lecture 1 exercises

Advanced Python - Spring 2022

Introduction

In this exercise set, we will use the concepts covered in the first lecture. Specifically, you will need to make use of *for* and *while* loops, *if*, *else*, and *elif* statements, and the *int*(), *pop*(), *print*(), and *input*() methods. You will also need to create your own methods with arguments and return values.

The exercise will be marked as OK if you get 11 / 21 points or more. Points are only awarded for exercises where your code produce the expected result, and where you provide comments describing what the code does.

If you have any questions, send an e-mail to *sigurd.alnes@inf.unibe.ch*. You should describe what you have done so far, and exactly why you believe you are stuck on a specific exercise.

Questions that do not ask for code should be answered as comments.

Exercises must be handed in via e-mail to *sigurd.alnes@inf.unibe.ch*. Deliver your submission as a compressed file (zip) containing one .py file for each main exercise (exercise_1.py, exercise_2.py, etc.). Use comments to indicate which sub-task your are answering within the main exercise (# Exercise 1a, etc.).

Name the zip archive according to the following format: **lecture-X_group-ID.zip**
Where X is the lecture number indicated in the title of this PDF, and ID is the ID of your group.

The exercises must be handed in by two students working together. If you do not have someone to collaborate with, please refer to *this* document to find another student without a group. If all groups have two members, add your information to an empty row, and preferably also create a forum post on Ilias announcing that you are looking for someone to collaborate with.

Collaboration outside the group (i.e. submitting the code of other groups as your own) will result in 0 points for the plagiarized exercises.

Deadline: 16:00, March 3.

Exercises

1. **Fix the code blocks** ..... 1 point

   (a) **Height difference** ..... 0.5 point
   The tallest person in recorded history, Robert Wadlow, was 272 cm tall. The code below should return different responses depending on the height of the user relative to Mr. Wadlow, amongst other how much shorter they are than Mr. Wadlow were. Fill in the "..." to complete the code.

```
wadlow_cm  = 272
user_cm = 179
if ... > ...:
    print('You are ', ... - user_cm, ' cm lower.')
elif wadlow_cm < ...:
    print('I don\'t believe you.')
else:
    print('...')
```

   Try out different values for *user_cm* and check the output.

   (b) **Indentation** ..... 0.5 point
   A fellow student wrote a program that uses a *for* loop to print only the numbers in the list *num_list* that are above or equal to 50, and otherwise prints "Too low". However, the student forgot to add indentation. Fix the the code below so that it runs as intended.

```
num_list = [ 32, 99, 12, 17, 77 ]
for num in num_list:
if num >= 50:
print(num)
else:
print('Too low')
```

2. **Variable assignment** ..... 3 points

   (a) **List** ..... 0.5 point
   What will be printed when you call *print*(*a*)?

```
a = ['x']
b = a
b += ['y']
print(a)
print(b)
```

   (b) **Explanation** ..... 1.5 points
   Explain why you got that answer on Exercise 2a.

(c) **String** ..... 1 point

Try running the code below and explain why the behavior is different from Exercise 2a.

```
a = 'x'
b = a
b += ' y'
print(a)
print(b)
```

3. **Loops** ..... 2 points

(a) **Sum the numbers** ..... 1 point

Use a *for* loop to sum up the numbers in the list *num_list* and print the sum.

```
num_list = [ 32, 99, 12, 17, 77 ]
```

(b) **Sum the numbers** ..... 1 point

Implement a *for* loop that counts the number of elements in *num_list* from exercise 3a, then calculate and print the mean of *num_list*.

4. **User interaction** ..... 2 points

Write a code snippet that asks for a persons name, and prints back "Hello, 'name'", where 'name' is the name provided by the user. If the user does not provide a name (just presses enter), the question should be asked again.

5. **Summing up numbers entered by user** ..... 1 point

Complete the below code. The user should be able to provide numbers that are added together in the variable *the_sum* until the loop is terminated (by entering "0"). After terminating the loop, the sum of the entered numbers should be printed when calling *print*(*the_sum*).

```
the_sum = 0
in_numb = None
while not in_numb == 0:
    ...
    ...
print(the_sum)
```

6. **PingPong** ..... 2 points

Write a code that iterates over all numbers from 0 to 99, including 0 and 99. For multiples of four, print 'Ping'. For multiples of seven, print 'Pong'. For numbers which are multiples of both four and seven, print "PingPong".

7. **Methods** ..... 10 points

(a) **Find smallest number in list** ..... 3 points

Complete the below code, such that you have defined a **method** that given a list, such as *num_list* defined in exercise 1b, returns the smallest number in the list. You are not allowed to use the built in *min*() method to solve this exercise.

```
def get_min(list):
    min_num = None
    ...
    ...
    return min_num
```

(b) **Find index of element in list** ..... 3 points
Write a **method** named *get_index*() that takes two arguments: a list and a variable (which can be an integer, a float, a string, etc.). Inside this method, find the index of the provided variable in the list and make that the return value of the method. If the variable does not exist in the list, return -1. You are not allowed to use the built in *index*() method to solve this exercise.

(c) **Remove smallest value from list** ..... 3 points
The *pop*() method removes an item at a specified place in a list, and returns that item, such as this:

```
num_list = [ 98, 67, 26, 99, 89, 12 ]
my_num   = num_list.pop(2)
print(my_num) # Should print: 26
```

In this exercise, write a code that:

i. Finds the minimum value in the list *num_list*, using your *get_min*() method from Exercise 7a.
ii. Gets the index of this minimum value, using your *get_index*() method from Exercise 7b.
iii. Uses *pop*() to remove the number at the index found in step 2. from the list *num_list*, and prints the return value of *pop*()

(d) **Sum smallest numbers until sum reaches 100** ..... 4 points
In this exercise, you should make use of the methods you implemented in Exercise 7a and 7b. Create a variable *sum* that starts off with the value 0. Write a *while* loop that **for each iteration**:

i. Finds the smallest value in the list *num_list* from Exercise 7c.
ii. Adds this smallest value to your variable *sum*.
iii. Removes this smallest value from your *num_list*.
iv. Checks if *sum* is greater than 100. Stops running if this is true.