Exam in Introduction to programming

2019-01-09

Instructions. Answer the questions below. They are roughly ordered by difficulty, but independent. Some questions are broken down into subquestions, *e.g*, question 4 might consist of subquestions 4-1, 4-2, and 4-3; the subquestions are supposed to be be treated in that order.

Provide your answers by editing the files provided in the `answers-xxxx` directory. You are not supposed to create any new files, nor change the names of the provided files in that directory.

Before handing in, replace the `xxxx` in the `answers-xxxx` directory with your ITU student ID, *e. g.*, `answers-jegp`. Hand in the renamed `answers-xxxx` directory as a single zip file called `answers-xxxx.zip`.

Some questions require extra input data. Such data is provided in a directory of the same name as the question. For instance, data for question four would be in the directory called `4`. In the questions below, the file name is given relative to the program you are supposed to write, so the data file `names.txt` for question four would be referred to as `../4/names.txt`, assuming you are standing in the `answers-xxxx` directory.

Any free text, such as program comments or explanations, can be written in Danish or English, but English is preferred.

At most 100 points can be earned in this exam.

Question 1 (6 Points)

For each of the following expressions, give their result.

```
13 + 2 + 3
3 * 12 + 1
2 ** 3
5 // 2
not (True or False)
2 * 3
"2" * 3
```

Question 2 (6 Points)

What is printed?

```
d = { 'Wonder Woman': 'DC', 'Catwoman': 'DC', 'Phoenix': 'Marvel' }
a = range(1, 10)
s = 'Lois Lane likes Catwoman'
print(len(a))
print(a[3])
```

```
print(s[a[0]])
print(s[5:8])
print(d[s[s.find('C'):]])
```

Question 3 (12 Points)

What is the purpose of the following code?

```
elements = []

for i in range(len(L)):
    if L[i] > 1000:
        elements.append(i)
```

Pick exacly one of the following:

a. Find all elements in `L` that are more than 1000 characters long.

b. Check if `L` contains a string that is more than 1000 characters long.

c. Find the indices of all elements in `L` that are greater than 1000.

d. Decide if `L` has more than 1000 elements.

e. Compute the sum of the elements of `L` that are greater than 1000.

Question 4 (12 Points)

The Finance Department got it all wrong again! For them, cash *inflow* is given as negative numbers, while cash *outflow* is given as positive numbers. However, the IT system wants it the other way around! For example, if the list of cash flow from the Finance Department is `L = [-3, -4, 2, 2]`, the list for the IT system is supposed to be `[3, 4, -2, -2]`.

I have come up with the following code to correct this problem:

```
def change_sign(L):
    for x in L:
    x = -x

L = [-3, -4, 2, 2]
change_sign(L)
print(L)
```

It doesn't work! It doesn't even run! Fix the code. (You are only allowed to change the body of the function.)

Question 5 (12 Points)

Consider the following function described only by a docstring:

```
def all_title_case(L):
    """Given a list L of strings, returns True if all strings in L
    are in title case and returns False otherwise.
    """
```

By "title case" we mean that each word in the string starts with an uppercase letter, and all other letters are lowercase. For instance, "This Is A String In Title Case" is in title case, while none of the three strings "TGIF", "Alice likes Bob" and "ScrollBar" are in title case.

5-1 (4 Points)

Provide two example lists `L1` and `L2` such that `all_title_case(L1)` should return True and `all_title_case(L2)` should return False (according to the docstring).

5-2 (8 Points)

Implement the function `all_title_case`.

Remark: Checking that a string has title case can be done with the Python string method `istitle()`. For example, `"Exam Question".istitle()` returns True. You are allowed to use this function (but you don't have to.)

Question 6 (12 Points)

You are running a meetup system in which people share their interests. A basic routine in such a system is to find common interests of two people. Your task is to define a function `find_common` that takes two lists `L1` and `L2` as arguments and returns a list of all the elements that are present in both `L1` and `L2`. You may assume that elements of the individual lists are unique, *i.e.*, there are no duplicates in `L1` and there are no duplicates in `L2`.

For example, if `L1 = ['Dancing', 'Computers', 'Rowing']` and `L2 = ['Computers', 'Books', 'Movies']`, the function has to return `['Computers']`. If `L1 = ['Fishing']` and `L2 = ['Swimming']`, the expected result is `[]`.

Question 7 (12 Points)

Write a class `TreasureChest` for your next role-playing game "Alice's adventures in GBI world". A treasure chest contains a valuable item, such as "a book of immense Python knowledge," "a scroll of critical thinking," or "the guardian of group work." When created, a treasure chest is locked, and its description should just be "a locked treasure chest." A chest has a method `open()`. After being opened, the description of the chest is its item.

Implement the class, including a constructor, the `open()` method, and a method that allows a description of the item to be printed. (For instance, you can implement a `print_description` or `get_description` method, or just override `__str__`.)

Create two treasure chest objects, and demonstrate that their description changes after they are opened. Here is an example of the intended behaviour:

```
>>> p = TreasureChest("a book of immense Python knowledge")
>>> q = TreasureChest("the guardian of group work")
>>> p.print_description()
a locked treasure chest
>>> q.print_description()
a locked treasure chest
>>> p.open()
>>> p.print_description()
a book of immense Python knowledge
>>> q.print_description()
a locked treasure chest
```

Question 8 (14 Points)

Write a billing system for a restaurant. The restaurant provides a file containing its menu. An example of such a menu is given as `../8/menu.txt`. In the file, each line consists of the name of a dish and its price, seperated by a comma. You may assume that the price is an integer and no dish contains a comma in its name.

You want to produce a bill for orders that are given as files. You can find two examples of such orders as `../8/order1.txt` and `../8/order2.txt`. Each line contains an item from the menu that has been ordered.

8-1 (8 Points)

Define a function `compute_total(menufile, orderfile)` that computes the total price the customer has to pay for the order given by its `orderfile` according to the menu given as `menufile`. The total price is returned. You may assume that every item in `orderfile` is actually present in the menu. You may assume that no item appears twice in the menu.

Example: The total price for the orders in `../8/order1.txt` is 112. Thus, `compute_total("../8/menu.txt", "../8/order1.txt")` has to return 112.

8-2 (6 Points)

Define a function `print_bill(menufile, orderfile)` that prints the bill in a nice format. Menu items should be sorted by name. Moreover, each menu item must the information how often it was ordered, and the total price for this menu item.

Example output: The result of `print_bill("../8/menu.txt", "../8/order1.txt")` could look like this:

```
1x Bread, 10
1x Egg, 32
2x Fish, 70

Total: 112
```

Question 9 (14 Points)

Consider the list

```
L = [1, 3, -1, 10, 11]
```

The number `3` is called the *median* of `L` because there are exactly two values less than 3 (namely, `1` and `-1`) and two values greater than 3. Namely, `10` and `11`. (Note that the median is *not* the average, *nor* the element in the middle of the list.) More generally, an element `x` is a *median* of a list `L` of numbers if the number of list elements in `L` less than `x` equals the number of elements in `L` greater than `x`.

(To make our life easier, let's assume that the length of `L` is odd and that it contains no two equal elements.)

Here is a simple program to find the median:

```
def median(L):
    while len(L) > 1:
        MIN = min(L)
        MAX = max(L)
        L.remove(MIN)
        L.remove(MAX)
    return L[0]

print(median([1,3,-1,10,11]))
```

9-1 (4 Points)

Which statement best describes the behaviour of the program:

a. Successively remove the largest and smallest elements from the list until only one remains, which must be the median of the original list.

b. Recursively remove the median until the maximum element equals the minimum, which must be the median.

c. Remove the first item from the list whose value is the median. It is an error if there is no such item, otherwise we found the median.

d. Remove the slice between the minimum and maximum elements in the list. If the list has length 1, it must be its own median.

9-2 (2 Points)

Run the program on some larger lists. What is the length of the largest list that the program can handle in a few seconds?

a. 1001

b. 10001

c. 100001

d. 1000001

9-3 (4 Points)

What is the running time of the program:

a. constant

b. logarithmic in the length of the list

c. linear in the length of the list

d. quadratic in the length of the list

9-4 (4 Points)

Write a program to find the median that is much faster. It must be able to find the median of any list of length 100000001 before you turn 100 years old.