Exam in Introduction to programming

2021-01-30

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 treated in that order.

Provide your answers by editing the files inside 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`, where xxxx is again your student ID.

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.

Declaration. **Your solutions and answers must be made by you and you only.**
This applies to program code and explanatory text that answer the exam questions.
You are not allowed to create the exam solutions as group work, nor to consult with fellow students, pose questions via internet fora, chat, SMS, phone, mail or the like.
You are allowed to communicate with SAP and the course teacher in case you believe there is a problem with exam questions; see the guidelines about this.

By submitting your solution to learnIT, you agree to the following sentence: "I hereby declare that I have answered these exam questions myself without any outside help." where "I" refers to the person that handed in the solution.

Points. At most 100 points can be earned in this exam.

Question 1 (7 Points)

For each of the following expressions, give their result.

```
4 + 2 + 3
str(4) + '2' + '3'
23 - 1 * 2
3 ** (3 - 1)
not (True or False)
True and True or False
3 * '3'
```

Question 2 (8 Points)

What is printed by the following 8 print statements?

```
numbers = [1, 2, 3, 4]
cities = {'Copenhagen': [55.676111, 12.568333],
          'Aarhus': [56.15, 10.216667],
          'Aberdeen': [57.15, -2.11],
          'Nuuk': [64.175, -51.738889]}
msg = '1. Test Introduction to Programming'
print(len(numbers))
print(numbers[2])
print(numbers[-2])
print(cities['Nuuk'])
print(cities['Copenhagen'][1])
print(msg[0] + msg[0])
numbers.append(list(cities.keys()))
print(len(msg))
print(msg[numbers[3]])
```

Question 3 (8 Points)

Consider the following code that describes a function that is supposed to work on a list `L` of integers:

```
def f(L):
    n = len(L)
    i = 0
    cnt = 0

    while i < n:
        if L[i] < 0:
            cnt = cnt + 1
        i = i + 1
    return cnt
```

3-1 (2 Points)

Provide a list `L` such that `f(L)` returns 3.

3-2 (6 Points)

Rewrite this code such that the `while` loop is replaced with a `for` loop.

Question 4 (6 Points)

What is the purpose of the following program?

```
data = 'Call me Ishmael.'
```

```
freqs = {}
for el in data:
    if el in freqs.keys():
        freqs[el] += 1
    else:
        freqs[el] = 1

m = (None, 0)
for c, f in freqs.items():
    if f > m[1]:
        m = [c, f]
print(m[0])
```

Pick exactly one of the following:

a. Check whether there is a string `c` in `data`.

b. Count how many numbers in `data` are lower than `el`.

c. Convert all elements in `data` from `keys` to `items`.

d. Print the character of `data` that occurs the most often.

e. Print the value `(None, 0)`.

Question 5 (12 Points)

Complete the following program by replacing the holes (`...`) with code as indicated in the comments. Don't worry about coming about with actual author names, i.e., fantasy names are okay.

```
# Make a list L of dead author names of length 3 or 4
L = ['Agatha Christie', ...
# print the length of L (should be 3 or 4)
print  ...
# insert 'Plato' at the end of the list
L...
# Make another list of length at least 3
Q = ['Goethe', ... ]
# Add all the authors in Q to L, using a single method (not a loop)
L...
# Remove the 2nd and 3rd authors from the list L using slicing
...
# Print each author in L on a separate line, without any quotation marks
...
```

Question 6 (15 Points)

You have noticed that many people in your company use passwords that are easy to guess. You made up the following rules.

1. A password has to contain at least 8 characters.
2. A password has to contain at least one lower-case letter, at least one upper-case letter, and at least one digit (0-9).

Examples: `1234` isn't a valid password at all and `secretpassword` only satisfies the first rule. The password `s3cRetp4ssword` has the properties of both rules.

Of course, nobody cared about adopting your rules. That's why you decided that everybody has to run their password through your validity checker.

6-1 (6 Points)

Write a function `is_valid_naive` that takes as argument a string `password`. It returns `True` if the password contains at least 8 characters, and returns `False` otherwise. (This means that only the first rule is implemented.)

6-2 (6 Points)

Write a function `is_valid` that takes as argument a string `password`. It returns `True` if the password has all of the properties stated in 1. and 2. above; otherwise it returns `False`.

Depending on how you implement your solution you might want to use the values of the variables `ascii_lowercase`, `ascii_uppercase`, or `digits` from the `string` module.

You can use the following code to get an idea how this will be helpful:

```
import string

print(string.ascii_lowercase)
```

6-3 (3 Points)

Write three `assert` statements that test the correctness of your implementation. At least one of them has to check a valid password, and at least one has to check an invalid password.

Question 7 (12 Points)

Make a class `FruitStand` that models a *fruit stand* according to the following description:

The class has three variables that model that a fruit stand has a certain *amount* of (unspecified) fruit, a certain amount of *money* as cash, and stands at a certain *location*. For example, the stand can start with 20 fruit, 50 money and can be located 'next to ITU'. The class is supposed to have four different methods: An *output* method that prints all available information about the stand (money, fruit count, location), a *sell* method that takes an amount of fruit to be sold (one fruit is sold for 5 money), a method to *receive* a fruit delivery (one fruit is bought for 2 money), and a method to *change the location* of the fruit stand.

It is not necessary to verify sensible use of the class (for instance, that the stand's number of fruit or money is always positive). It is necessary to update the variables containing money and fruit count after selling/receiving fruit.

Instantiate one object from the class and call the methods to show that the stand can sell fruit, receive fruit deliveries, and change location.

Make sure to add explanatory docstrings as documentation to *all* methods of your class.

Question 8 (16 Points)

Københavns Kommune put you in charge to run their contract tracing program for COVID-19 cases.

In the files `../8/1.txt` and `../8/2.txt` you find data about which people met within the last few days. For example, `../8/1.txt` looks like this:

```
Cortez Georgann
Cortez Shaniqua
Kendall Georgann
```

This means that Cortez had direct contact with two other people: Georgann and Shaniqua. Moreover, Georgann also met with Kendall.

8-1 (6 Points)

Write a function `had_contact` that takes two arguments: the name of a person, and the path to a file that contains the current contact information. The function returns `True` if the person was in direct contact with another person according to the contact information, and `False` otherwise. For example, `had_contact('Cortez', '../8/1.txt')` has to return `True`, `had_contact('Theodora', '../8/1.txt')` has to return `False`.

8-2 (4 Points)

Write a function `list_contacts` that takes the same two arguments as above: the name of a person, and the path to a file containing the current contact information. The function returns a list of all people that the person had direct contact with. (The order doesn't matter.) Make sure that this list does not duplicate entries, because the same pair of people could have met several times.

8-3 (6 Points)

Unfortunately, testing capacities are at their limits and not everyone can get tested. In the file `../8/risk.txt` you find the risk levels that indicate how severe an infection would be to a person as integers ranging from 0 (no risk) to 9 (high risk).

Write a function `list_contacts_at_risk` that takes two additional arguments to the ones above. The third argument is an integer `risk`, representing the level at which a person has to sign up to get tested. The fourth argument is the path to a file that contains the risk information, e.g., `../8/risk.txt`. Your function has to return all the contacts of a person that are at least at risk level `risk` according to the risk information file.

For example, let's say we provide Cortez as the person to list contacts. As above, Georgann and Shaniqua have been in contact with Cortez. In `../8/risk.txt` Shaniqua has a risk of 4 and Georgann is at risk 1. So, if the provided `risk` level is 1 or smaller, then both Georgann and Shaniqua have to be reported, if it's between 2 and 4, then only Shaniqua has to be reported, and if it's 5 or above, noone is reported.

Question 9 (16 Points)

Under COVID-19 we are all getting used to keeping our distance. In the following task, we will consider the scenario that someone is telling us the positions of all chairs in the room. Each chair's position is given by a pair (x, y) stored as a list with two elements. The question we have to answer is whether there are two chairs that are closer than 1.5 meter apart from each other.

The following code is supposed to solve this task.

```python
import math

def is_close(chair, other_chair):
    """Computes the distance between two points (x1, y1) and (x2, y2) using the
    formula (x1-x2)^2 + (y1-y2)^2 and returning the square root of this."""
    distance = (chair[0] - other_chair[0])**2 + (chair[1] - other_chair[1])**2
    distance = math.sqrt(distance)
    return distance <= 1.5

def has_close_seats(chair_list):
    """Given a list of chairs with their positions [x, y], returns True if there
    is a pair of close chairs, and False otherwise."""
    for i in range(len(chair_list)):
        for j in range(i+1, len(chair_list)):
            if is_close(chair_list[i], chair_list[j]):
                return True
            else:
                return False

chair_list1 = [[0, 0], [10, 0], [20, 0]] # no close pair, should return False
print(has_close_seats(chair_list1))

chair_list2 = [[1, 1], [3, 2], [0, 1]] # has a close pair, should return True
print(has_close_seats(chair_list2))
```

Don't worry about the intricate math to compute the distance in the function `is_close`. You probably saw it in high school, but the details don't matter when solving this task.

9-1 (4 Points)

For some reason, this function does not work. (For example, it returns `False` on the input `[[1, 1], [3, 2], [0, 1]]`, although the first and third chair are clearly only 1 meter apart. Correct the implementation.

9-2 (4 Points)

Run the program on some larger inputs. (Don't worry about inventing entertaining content. Just take a list like [[1, 0], [3, 0], [5, 0], ...].) What is the largest list length that the corrected program can handle in no more than a minute?

a. 50

b. 5000

c. 500000

d. 50000000

9-3 (4 Points)

What is the running time of the corrected program?

a. constant

b. linear in the length of the list

c. quadratic in the size of the list

d. exponential in the size of the list

9-4 (4 Points)

Add a function `ensure_proximity(chair_list)` to your program. The function modifies a given list of chair positions in such a way that the returned list of chair positions does not have any pair of chairs that are too close. That is, given the list of chair positions for which there *might be* chairs that are too close to each other, remove some chairs such that the returned list of chair positions does not have a pair that is too close. (A naive solution that just returns the empty list or always only a single chair gives 0 points.)

Please motivate the idea behind your approach by writing a few comments.