PHOT 110: Introduction to programming

Final exam questions, version A

Michaël Barbier, Spring semester (2023-2024)

Before you start

The final exam counts for 50% of your total grade of the course. You will get 3 hours to complete your exam. The exam is performed on the computer, and you do not need to provide any answers on paper. There is one question (the third question) which asks to correct a script which contains various errors, other questions require you to write Python scripts.

During our final exam, there will be a very short (about 5 minutes) oral part. This part is only about the projects, and impacts only the grade of your projects. You will be asked a couple of short questions about the project: the code you wrote and/or the accompanying report. If required you can prepare the answer on paper (there will be paper available), you also will have a print-out version of your report. In principle, you do not need to prepare for this oral part, the questions' only purpose is to see whether you understood what you did in the project.

Take the following points into account before you start the exam.

- For those who use their own computer/laptop: you do not need internet, we will ask you to switch of your WiFi before the exam starts. The exam files are distributed via USB drive. Copy all files from the USB drive to your computer before you start:
  - The questions: `phot110_exam_questions_x.pdf`
  - The script with errors for question 3: `script_with_errors.py`
  - Three cheat-sheets: for Python, Numpy, and Matplotlib.
- For those who work on the desktop computers in the lab: all the required files should be found on the Desktop.
- Make sure that the `Numpy` and `Matplotlib` libraries are installed already (we will test this together before the exam starts).
- At the end of the exam we will go around with USB drives to collect the exams: put your solutions (and output files) in a folder which has both your name and your student number in the folder name, for example "Michael_Barbier_30029034".

- Let us know if during the exam there is any issue with the computer, PyCharm, or libraries. We will try to verify this up front, but please inform us in case of any issues.

You will be asked to save plots to your folder, this can be done using the `savefig` method, see the following example (the output of this example is a simple line plot):

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
x = [1, 4, 3, 0]
y = [2, 2, 5, 4]
ax.plot(x, y)
fig.savefig("output_plot_example.png")
```

Questions

Please solve all of the following 7 questions. Each question is worth an equal amount of points (out of 100).

Question 1:

Write a script that prints the cumulative values of the series $\sum_{n=1}^{N} n$ up to the Nth term:

$$\sum_{n=1}^{N} n = 1 + 2 + \cdots + N$$

Use a loop (e.g. `for` or `while`) structure to print the cumulative values for increasing `N` on different lines. Your solution script should print the cumulative values up to order `N = 5`, and the output should look similar to:

```
Cumulative sum of first 1 terms: 1
Cumulative sum of first 2 terms: 3
Cumulative sum of first 3 terms: 6
Cumulative sum of first 4 terms: 10
Cumulative sum of first 5 terms: 15
```

Save your solution as a script with file name: `solution_1.py`.

Question 2: New password

Make a script that prompts a user repeatedly to create a new password until he/she provides a valid one. For the password to be valid, it should contain at least `N` characters and not contain any spaces. If the number of characters provided by the user is less than `N`, tell the user that the input is invalid and prompt the user again. Stop the program when a correct password is provided. This is example output of a correct script (with a character limit `N = 6`):

```
Please provide a password (at least 6 characters, without spaces): Abc4
The provided password is not valid! Please try again.

Please provide a password (at least 6 characters, without spaces): Abc 4567
The provided password is not valid! Please try again.

Please provide a password (at least 6 characters, without spaces): Hello2345
Your password has been successfully changed!
```

Save your solution as a script with file name: `solution_2.py`.

Question 3: Correct a Python script

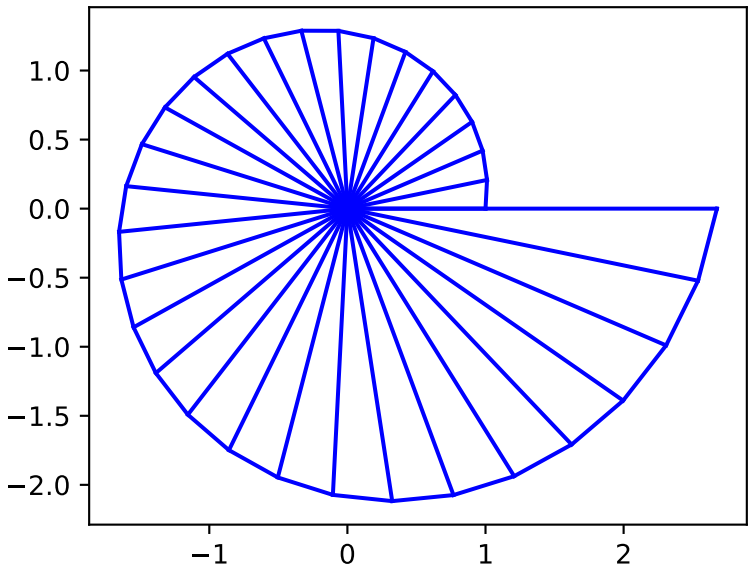
Open the script with name: `script_with_errors.py` and correct the errors. Save the corrected script with file name `solution_3.py`.

The corrected script should plot a round staircase-like geometry existing of triangles. The N triangles are plotted by first calculating points at equidistant angles α_n and increasing radial distance (starting from 1) according to:

$$\alpha(n) = \frac{2\pi\, n}{N}$$
$$L(n) = \left(\frac{N}{N-1}\right)^n$$

With n an integer in interval $[0, N]$ and $N = 32$. This graph is then saved as a png-file with file name `output_script_with_errors.png` to the hard disk.

The output graph should look as the plot below:



Question 4: 2D density plot

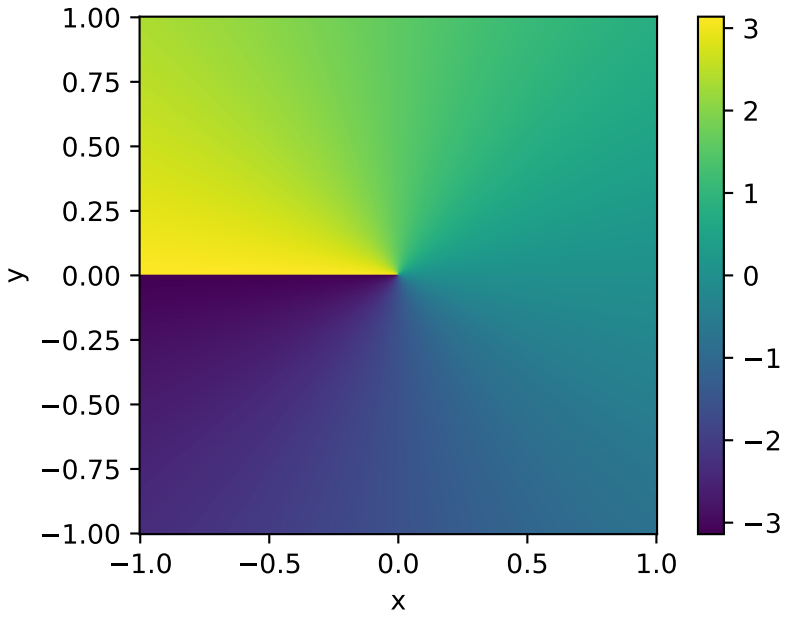
Plot $z(x, y)$ given by the (anti-clockwise) angle $\phi \in [-\pi, \pi]$ that the coordinates (x, y) makes with the positive x-axis:

$$z(x,y) = \begin{cases} \arctan(y/x) & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}$$

whereby you can use the arctangent function of Numpy: please use the version with two arguments called `arctan2(y, x)` available in Numpy (argument `y` is before argument `x`). This function will take care of the division-by-zero issue. Plot the resulting function $z(x, y)$ as a density plot. Remember that you can make a density plot (with a colorbar) using the commands:

```
fig, ax = plt.subplots()
pc = ax.pcolormesh(xx, yy, zz, rasterized=True)
fig.colorbar(pc, ax=ax)
```

For each of the functions you can use the same (x, y)-values within an 2D interval/domain with $x \in [-1, 1]$ and $y \in [-1, 1]$ (take a sufficiently fine grid of (x, y) coordinates to obtain a smooth density plot). **Save the plot** under the file name: `output_plot_angle.png`. Save your solution as a script with file name: `solution_4.py`. The output of the script should look as in the plot below:



Question 5: Creating and using modules

Create a **module** (a separate script file) with file name `module_wire.py` which helps to calculate some properties of electric wires (resistance of the wire and current through the wire). The module should contain two functions :

- `resistance(rho, L, A)`: Calculates the resistance of the metal wire which is a function of the wire's resistivity ρ, the length of the wire L, and its cross-sectional area A:

$$R = \frac{\rho\, L}{A}$$

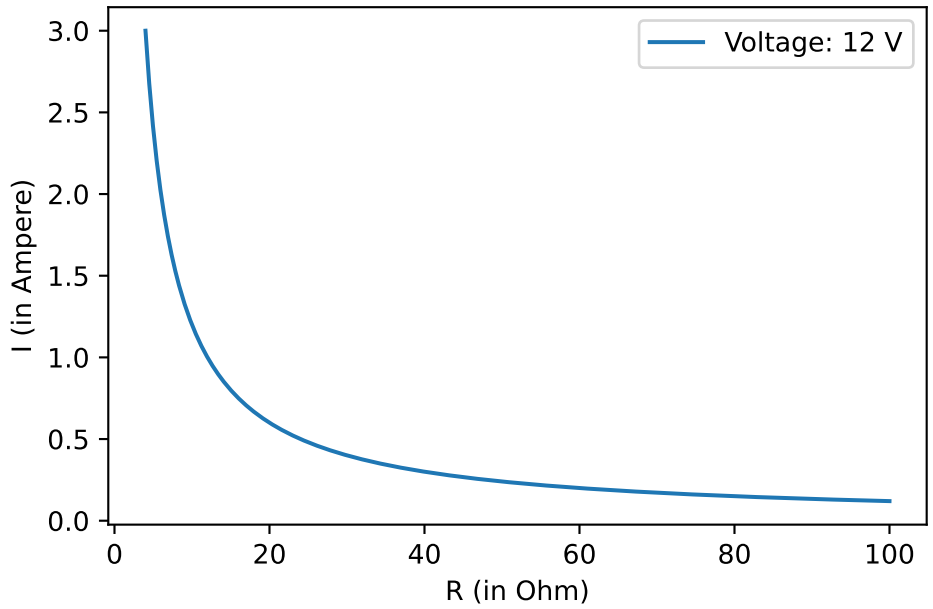
- `current(V, R)`: calculates the current through a wire with resistance R according to Ohm's law:

$$I = V/R$$

  with V the voltage over the wire.

Afterwards, import and use this module in a script (with file name: `solution_5.py`) that plots the current I (in Ampere) as function of the resistance $R \in [4, 100]$ Ohm, and take the voltage $V = 12$ Volt using the `current(V, R)` function from the module. Save the plot as a png-file with file name `output_plot_current.png` to the hard disk.

The output graph should look as the plot below:



Question 6: Dictionaries

Define the following dictionary with desserts and their prices (in TL) in your script:

```
menu = {
  "Apple pie": 14,
  "Chocolate cookie": 18,
  "Tiramisu": 21,
  "Cheesecake": 25,
  "Ice cream": 15
}
```

Then let the script automatically print out the desserts that cost 20 TL or less. The output of a correct script should be the following:

```
Within a budget of 20 TL we can offer the following desserts on the menu:
  - Apple pie for 14 TL.
  - Chocolate cookie for 18 TL.
  - Ice cream for 15 TL.
```

Save your script under the name `solution_6.py`.

Question 7: Classes

Create a class named `Taxi` which has two attributes

- `name`: the name of the driver,
- `total_earnings`: earned money so far (initialized to 0)

The Taxi object represents a taxi driver that earns money when giving passengers a ride to their destination. Give the class a constructor that accepts one argument (in addition to the mandatory `self` argument), the drivers name: `__init__(self, name)`.

Then add the following method:

- `ride_price(self, distance)`: a method which calculates (and returns) the taxi fare (in TL) for a ride using the formula:

$$\text{taxi fare (in Turkish lira)} = 10 \times \text{distance} + 25$$

  where the cost per kilometer is 10 TL and the base price is 25 TL. Add the taxi fare to the earned money so far represented by the attribute `total_earnings` as defined above.

Then use the class in your script to make a Taxi object for taxi driver "Joe". Prompt then the user to input a destination distance and reply with the taxi fare. This is example output of a correctly working script:

```
Hello Sir/Madam, please provide the distance (in km): 45
The ride will cost: 470 TL
```

Save your script under the name `solution_7.py`.