PHOT 110: Introduction to programming

Final exam questions, version A

Michaël Barbier, Spring semester (2024-2025)

Before you start

The final exam counts for 70% 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.

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

- The exam files are on the computers, please tell us if any file is missing/cannot be opened:
  - 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.
  - The data file for question 7: "FTIR_olive_oil.txt"
- Make sure that the `Numpy` and `Matplotlib` libraries are installed already.
- At the end of the exam **we will collect the exams**: put your solutions (and output files) and the task description 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: Loops

Write a script that prompts the user to input a word and then print every character of the reversed word on a separate line to the console. Use a loop (e.g. `for` or `while`) structure to perform this task. The output should look similar to:

```
Please provide a word or sentence: Computer
r
e
t
u
p
m
o
C
```

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

Question 2: Tables of multiplication

Create a script that repeatedly provides the user with a simple calculus question such as `5 x 7 = ?` to train the tables of multiplication, and verifies his/her answer. Generate the numbers (between 1 and 10), at random as exemplified in the following code:

```
import random

a = random.randint(1, 10)
print(a)
```

```
4
```

- Whenever the user provides a correct answer, provide a next question,
- When the user provides a wrong answer or gives invalid input, let him/her try again the same question.

- If the user writes the word `stop`, stop the program.

```
Tables of multiplication (write "stop" to stop exercising).
3 x 5 = ? : 35
Correct!
6 x 3 = ? : blah
The provided answer is not valid! Please try again.
6 x 3 = ? : 18
Correct!
1 x 8 = ? : 1
This answer is wrong!
1 x 8 = ? : 8
Correct!
4 x 2 = ? : stop
Goodbye!
```

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

Question 3: Correct a Python script

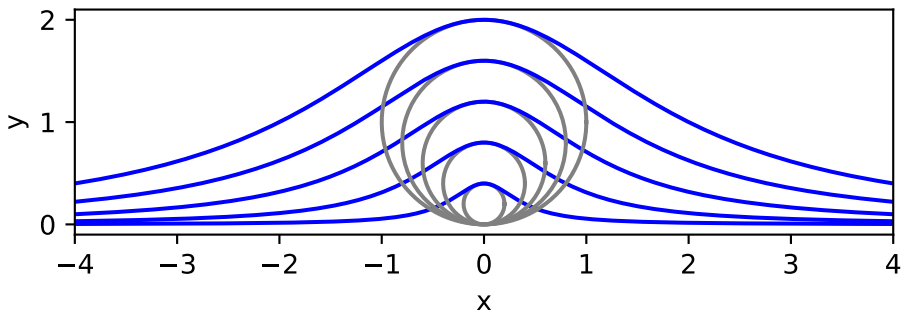
Open the script with name: `script_with_errors.py` and correct the errors. Save the corrected script with file name `solution_3.py`.

The corrected script should plot the "Witch of Agnesi" curve together with the corresponding circle:

$$y = \frac{8a^3}{x^2 + 4a^2}$$

The curve is plotted for various radii $a = 0.2, 0.4, 0.6, 0.8, 1$. 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 the intensity $I(x, y)$ of a diffraction pattern from a rectangular aperture with using the formula:

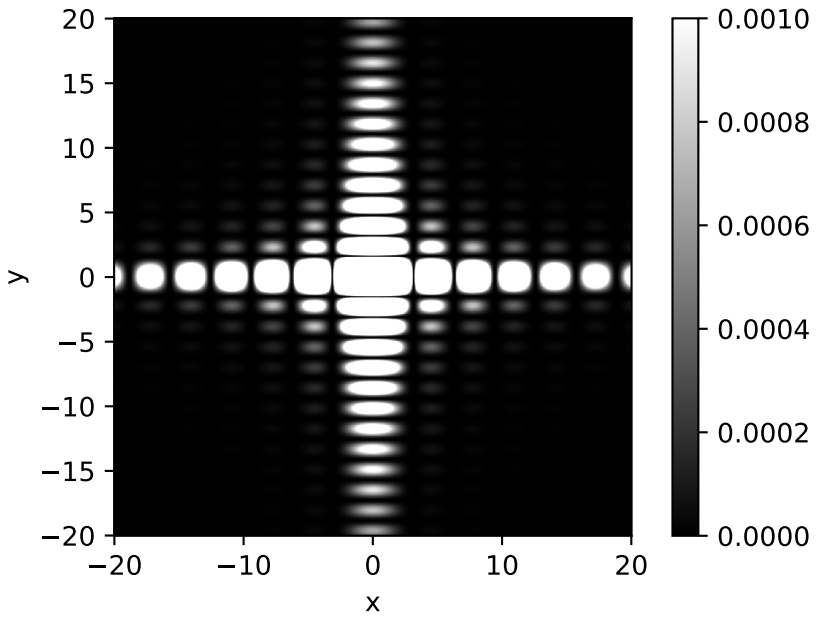
$$I(x, y) = \text{sinc}^2\left(\frac{x\,\delta y}{\pi}\right)\,\text{sinc}^2\left(\frac{y\,\delta y}{\pi}\right)$$

whereby you can use the Sinc function of Numpy called `sinc(x)` available in Numpy. Take parameters $\delta y = 1$ and $\delta y = 2$. Plot the resulting function $I(x, y)$ 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, vmax=0.001, cmap="gray")
fig.colorbar(pc, ax=ax)
```

Seeting the arguments `vmax=0.001` and `cmap="gray"` gives the same z-limit and colormap as in the output plot shown below.

For each of the functions you can use the same (x, y)-values within an 2D interval/domain with $x \in [-20, 20]$ and $y \in [-20, 20]$ (take a sufficiently fine grid of (x, y) coordinates to obtain a smooth density plot). **Save the plot** under the file name: `output_plot_sinc.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:



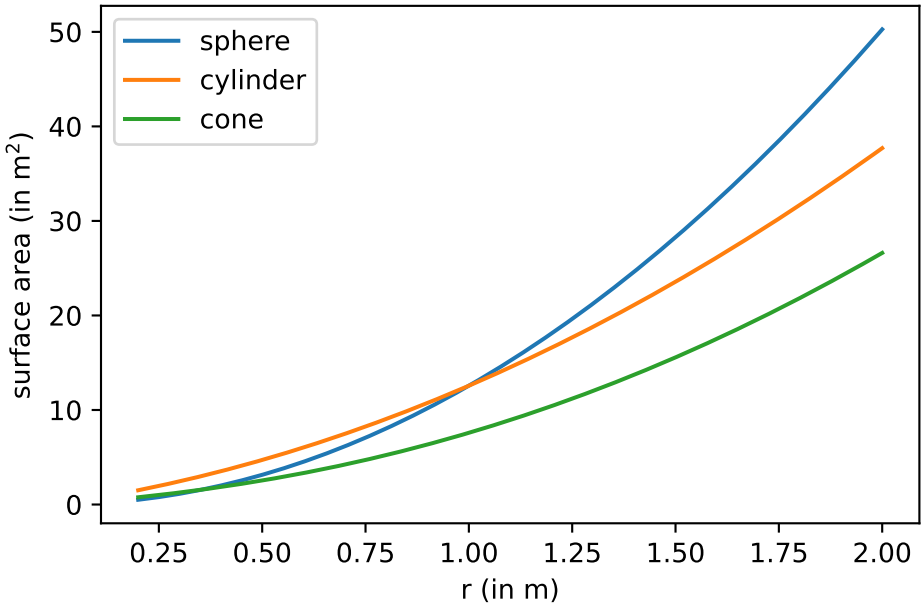
Question 5: Creating and using modules

Create a **module** (a separate script file) with file name `module_shapes.py` which helps to calculate areas of various geometrical shapes. The module should contain three functions:

- `surf_sphere(r)`: Calculates the surface area A of a sphere with radius r: $A = 4\pi r^2$
- `surf_cone(r, h)`: Calculates the surface area of a cone with radius r and height h: $A = \pi r^2 + \pi r \sqrt{h^2 + r^2}$
- `surf_cylinder(r, h)`: Calculates the surface area of a cylinder with radius r and height h: $2\pi r^2 + 2\pi r h$

Afterwards, import and use this module in a script (with file name: `solution_5.py`) that plots the surface area of a sphere, cone, and cylinder as function of the radius r in meter (putting the height $h = 1$ meter), by using the `surf_sphere(r)`, `surf_cone(r, h)`, and `surf_cylinder(r, h)` function from the module. Save the plot as a png-file with file name `output_plot_shapes.png` to the hard disk.

The output graph should look as the plot below:



Question 6: Dictionaries

Define the following dictionary with costs of four friends: Jack, An, Boris, and Nancy, who went on a trip ("Nancy" didn't pay anything yet). Use the following code in your script:

```python
costs = [
    {"cost": "dinner", "who": "Jack", "amount": 360},
    {"cost": "train", "who": "An", "amount": 1240},
    {"cost": "supermarket", "who": "Jack", "amount": 400},
    {"cost": "hotel", "who": "Boris", "amount": 4800},
]
```

Then let the script automatically calculate the total cost of the trip and the amount of money (in turkish lira) that each person still has to pay to (or should get from) the others. That is, the amount they already paid minus the total cost divided by four.

The output of a correct script should be the following:

```
Balance sheet:
  - Jack: -940.0 TL.
  - An: -460.0 TL.
  - Boris: 3100.0 TL.
  - Nancy: -1700.0 TL.
```

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

Question 7: Classes

Create a class named `Spectroscope` which has two attributes:

- `name`: the name or identifier of the device, e.g. "spectro1000xb" or "heatsensor-ST23",
- `data`: contains the measurement data (once loaded in). Initially it is `None`.

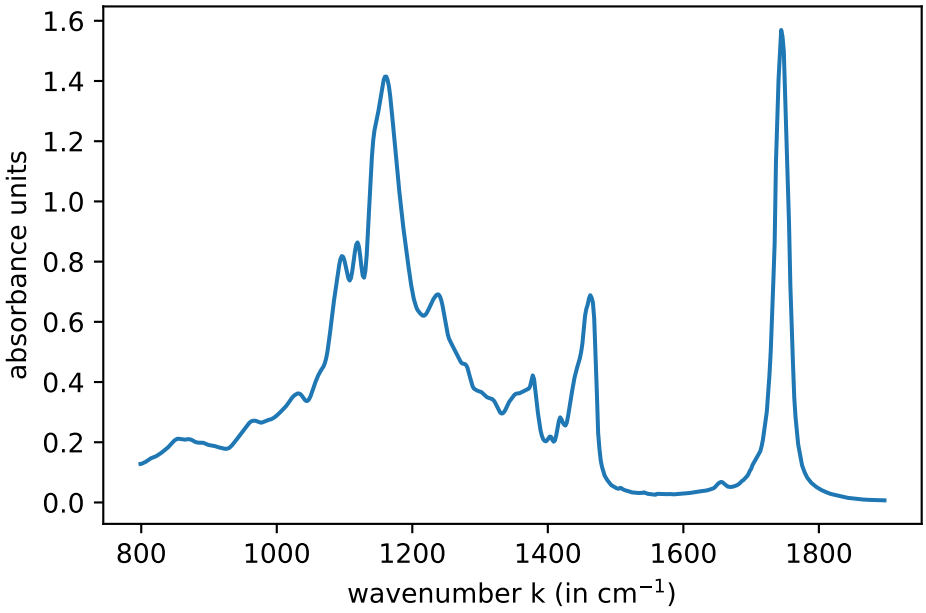
Give the class a constructor that accepts one argument (in addition to the mandatory `self` argument), the name (`name`) of the device: `__init__(self, name)`. Further create the following two methods:

- `load_data(self, data_file)`: which loads the spectroscopy data. The data is saved as a ".txt" file containing tab-separated values, organized in two columns (wavenumber and absorbance). This data can be loaded into a Numpy array using the `loadtxt(data_file)` function of Numpy.
- `plot(self)`: which visualizes the data by plotting the spectrograph (see the example output below).

Then use the class in your script to make a Spectroscope object with identifier "sx100", load, and plot the provided data (FTIR spectrum of virgin olive oil: "FTIR_olive_oil.txt") using the following code:

```
spectro = Spectroscope("sx100")
spectro.load_data("FTIR_olive_oil.txt")
spectro.plot()
```

The (correctly working) script should give the following output figure:



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