PHOT 110: Introduction to programming

Final exam questions and solutions, version A

Michaël Barbier, Spring semester (2023-2024)

Questions/problems and solutions

We first show the question, then the solution and then how the points are counted (raw score). The points of each question are normalized to 100/7. We then sum these for the 7 questions to get a total score on 100. If question i has M_i points and one obtained m_i/M_i points, then the total score S is:

$$S = \sum_{i=1}^{7} [\frac{100}{7} \times m_i/M_i]$$

In the score calculation of the solutions to the questions, we appoint different amount of points. However, every question has the same weight (100/7), due to the above mentioned normalization of the points.

Remark that there is a minimal amount of comments used in the problem solutions. This is to keep the code listings within this document more compact. In real scripts I would advice to put more comments to document your code.

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`.

Solution 1

```python
N = 5
s = 0
for n in range(1,N+1):
  s += n
  print(f"Cumulative sum of first {n} terms: {s}")
```

Score calculation

6 points to be obtained:

1. Being able to calculate the cumulative sums (1 point)
2. Having multiple lines (1 point)
3. Having the cumulative sum matching the number of terms (1 point)
4. Printing the resulting sentences in correct format (1 point)
5. Using a loop structure to print one sentence per line (1 point)
6. Script runs without or with only trivial errors (1 point)

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`.

Solution 2

```python
N = 6
while True:
    pw = input(f"Please provide a password (at least {N} characters, without spaces): ")
    if (len(pw) < N) or (" " in pw):
        print("The provided password is not valid! Please try again.")
    else:
        print("Your password has been successfully changed!")
        break
```

Score calculation

6 points to be obtained:

1. Prompt the user for input (1 point)
2. Verify the number of characters (1 point)
3. Verify whether the password includes any space symbols (1 point)
4. Print a message when the password is changed (1 point)
5. Prompt the user again if input is not appropriate (1 point)
6. Script runs without or with only trivial errors (1 point)

Question 3: Correct a Python script

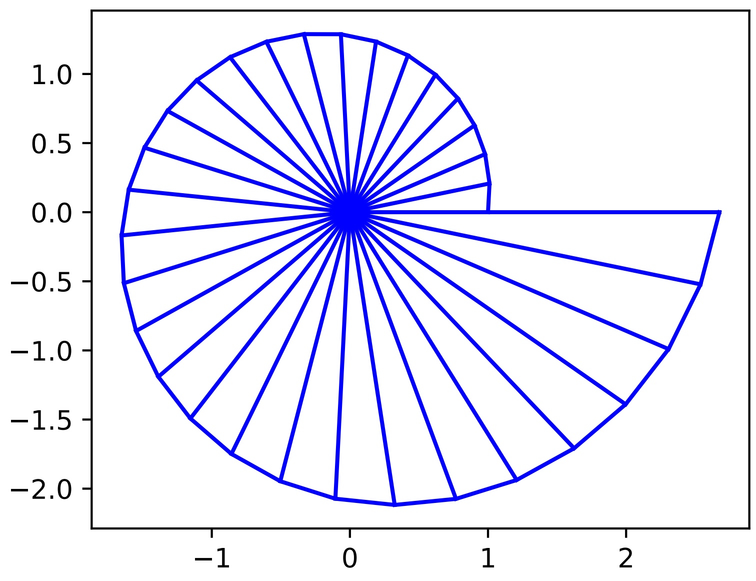
Open the script with name: `script_with_errors.py` and correct the errors. Save the corrected script with file name `solution_3.py`.

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:



Solution 3

Procedure to reach to the solution:

- On line 13: the `subplots()` returns two objects, the first is the figure handle: replacing the line by `fig, ax = plt.subplots()` will fix the issue.
- On line 18: `arange` is a Numpy function, replacing the line by `n = np.arange(N)` will fix the issue.
- Lines 20-21: Indentation should be removed.
- On line 24: variable `y` should be `ys`.
- On line 27: The line should be indented (for loop).
- On line 30: The property parameter should be a string: `ax.set_aspect(equal)` should be replaced by `ax.set_aspect("equal")`.

```python
# This is the corrected script.

# Load the necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Initialize the figure
fig, ax = plt.subplots()

# Determine the outer points of the "staircase"
N = 32
angles = np.linspace(0, 2*np.pi, N)
n = np.arange(N)
Ls = (N/(N-1)) ** n
xs = Ls * np.cos(angles)
ys = Ls * np.sin(angles)

# Plot the "outer border"
ax.plot(xs, ys, color="blue")
# Plot the radial lines
for x, y in zip(xs, ys):
    ax.plot([0, x], [0, y], color="blue")

# Set the axes aspect ratio
ax.set_aspect("equal")

# Save the resulting plot
fig.savefig("output_script_with_errors.png")
```

Score calculation

8 points to be obtained:

1. Define a figure/axis to plot in (1 point)
2. Having the angles and radii (1 point)
3. Having the x and y coordinates (1 point)
4. Plotting the outer border (1 point)
5. Plotting the radial lines (1 point)
6. Having the same plot style, axes with equal aspect ratio (1 point)
7. Saving the output to a file (1 point)
8. Script runs without or with only trivial errors (1 point)

Question 4: 2D density plot

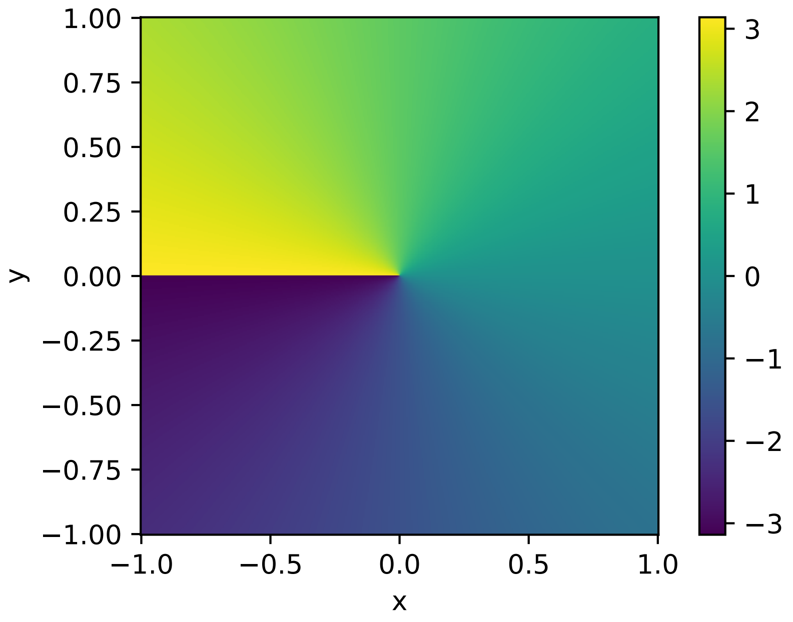
Plot $z(x, y)$ given by the (anti-clockwise) angle $\phi \in [-\pi, \pi]$ that the coordinates (x, y) makes with the positive x-axis:

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

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:



Solution

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

x = np.linspace(-1, 1, 400)
y = x
xx, yy = np.meshgrid(x, y)
zz = np.arctan2(yy, xx)

fig, ax = plt.subplots()
pc = ax.pcolormesh(xx, yy, zz, rasterized=True)
ax.set_aspect("equal")
ax.set_xlabel("x")
ax.set_ylabel("y")
fig.colorbar(pc, ax=ax)

fig.savefig("output_script_angle.png")
```

Score calculation

7 points to be obtained:

1. Creating multiple x and y values within the intervals forming the domain (1 point)
2. Creating 2D arrays of coordinates within the domain
3. Understanding how to apply the `arctan2(y,x)` function and obtaining correct z-values from chosen (x, y)-coordinates (1 point)
4. Having a smooth plot (1 point)
5. Style of the plot looks like the example (1 point)
6. Enabling saving the plot (1 point)
7. Script runs without or with only trivial errors (1 point)

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:

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

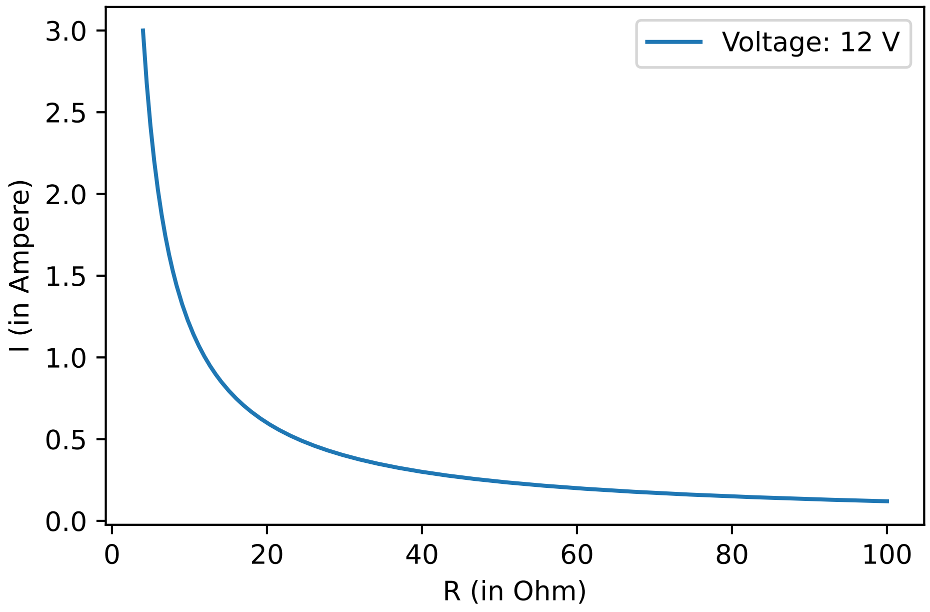
- `current(V, R)`: calculates the current through a wire with resistance R according to Ohm's law:

$$I = V/R$$

  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:



Solution

The solution exists out of two files:

- `module_wire.py`: the module providing functionality to calculate some electric properties of electric wires.

- solution_5.py: the main script which plots the current I (in Ampere) as function of the resistance R.

Remark: In the code listings I removed the underscores of the file names as I had a problem with rendering them in this pdf-file.

Listing 1 modulewire.py

```python
def current(V, R):
  return V / R

def resistance(rho, L, A):
  return rho * L / A
```

Listing 2 solution5.py

```python
import numpy as np
import matplotlib.pyplot as plt

# Import the module we created
import modulewire as mw

# Calculate current for resistance R in an interval
V = 12
R = np.linspace(4, 100, 400)
I = mw.current(V, R)

# Plot the current as function of the resistance
fig, ax = plt.subplots()
ax.plot(R, I)
ax.set_xlabel("R (in Ohm)")
ax.set_ylabel("I (in Ampere)")
ax.legend(["Voltage: 12 V"])
fig.savefig("output_plot_current.png")
```

Score calculation

10 points to be obtained:

1. Creation of a separate module file (1 point)

2. Implementing the correct expressions for the electric properties (1 point)
3. Function definitions implementing both electric properties (1 point)
4. Importing the module (1 point)
5. Creating x coordinates within the correct interval (1 point)
6. Calling the `current()` function to calculate the current as function of the resistance (1 point)
7. Having a smooth plot (1 point)
8. Enabling a similar output plot as in the question (1 point)
9. Saving the output plot to a file (1 point)
10. Script runs without or with only trivial errors (1 point)

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`.

Solution

```
# menu dictionary copied from question description
menu = {
  "Apple pie": 14,
  "Chocolate cookie": 18,
  "Tiramisu": 21,
  "Cheesecake": 25,
  "Ice cream": 15
}


# List all desserts within the budget
print(f'Within a budget of 20 TL we can offer the following desserts on the menu:')
for k, v in menu.items():
  if v <= 20:
    print(f'  - {k} for {v} TL.')
```

Score calculation

8 points to be obtained:

1. Understanding how a dictionary is written (1 point)
2. Having a dictionary similar to the one of the task description (1 point)
3. Using a loop to iterate over all desserts in the list (1 point)
4. Using a conditional clause for the price (1 point)
5. Printing the dessert names on separate lines (1 point)
6. Printing each dessert name together with its price (1 point)
7. Enabling similar output to the example (1 point)
8. Script runs without or with only trivial errors (1 point)

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: 475 TL
```

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

Solution

```python
class Taxi():
  def __init__(self, name):
    self.name = name
    self.total_earnings = 0

  def ride_price(self, distance):
    taxi_fare = 10 * distance + 25
    self.total_earnings += taxi_fare
    return taxi_fare

taxi = Taxi("Joe")
distance = float(input("Hello Sir/Madam, please provide the distance (in km): "))
print(f"The ride will cost: {taxi.ride_price(distance)} TL")
```

Score calculation

12 points to be obtained:

1. Creating a class (1 point)
2. Having all required class attributes (1 point)
3. Having a constructor which allows giving the taxi driver a name (1 point)
4. Having the method `ride_price(self, distance)` which calculates the fare for a ride (1 point)
5. The `ride_price(self, name)` method updates the `total_earned` and `total_distance` attributes (1 point)

6. The `ride_price(self, name)` method is correctly implemented (1 point)
7. Knowing how to create a Taxi object with driver name “Joe” (1 point)
8. Prompting the user for input (1 point)
9. Converting the distance to a float or integer (1 point)
10. Knowing how to apply the `ride_price` method (1 point)
11. Enabling similar output to the example (1 point)
12. Script runs without or with only trivial errors (1 point)