PHOT 110: Introduction to programming

Final exam example questions and solutions

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. 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 all the strings in a list of strings after appending them to the string: `"King Arthur is "`. Use a loop (e.g. `for` or `while`) structure to automate the process. Your solution script should print each sentence on a separate line. If you use the following list of strings:

```
words = ["the greatest", "a mythological figure", "some old knight"]
```

then the output should look similar to:

```
King Arthur is the greatest.
King Arthur is a mythological figure.
King Arthur is some old knight.
```

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

Solution 1

```python
words = ["the greatest", "a mythological figure", "some old knight"]
for word in words:
  print(f"King Arthur is {word}.")
```

```
King Arthur is the greatest.
King Arthur is a mythological figure.
King Arthur is some old knight.
```

Score calculation

6 points to be obtained:

1. Being able to print the outcome by string concatenation (1 point)
2. Having multiple lines (1 point)
3. Printing a sentence for each possible suffix (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: Cookie vending machine

Make a script that prompts a user repeatedly to input how many cookies he/she wants until the vending machine has no cookies left. For this, prompt the user to give a number between 1 and the number of cookies left (which you calculate). Tell the user each time how many cookies are left. If the number provided by the user is not a valid number (too large, too small, or not a number), tell the user that the input is invalid and prompt the user again. Stop the program when all cookies are finished. This is example output of a correct script (with a starting value of `N = 12` cookies):

```
How many cookies do you want to buy (12 left): 5
Here are your cookies! Have a good day.

How many cookies do you want to buy (7 left): 10
Sorry, your request is invalid. Please fill in a valid amount.

How many cookies do you want to buy (7 left): five
Sorry, your request is invalid. Please fill in a valid amount.

How many cookies do you want to buy (7 left): 7
Here are your cookies! Have a good day.

Unfortunately we are out of cookies!
```

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

Solution 2

```python
n_cookies = 12
while n_cookies > 0:
    try:
        n_buy_str = input(f"How many cookies do you want to buy ({n_cookies} left): ")
        n_buy = int(n_buy_str)
        if 0 < n_buy <= n_cookies:
            n_cookies = n_cookies - n_buy
            print("Here are your cookies! Have a good day.\n")
        else:
            print("Sorry, your request is invalid. Please fill in a valid amount.\n")
    except ValueError:
        print("Sorry, your request is invalid. Please fill in a valid amount.\n")

print("Unfortunately we are out of cookies!")
```

Score calculation

6 points to be obtained:

1. Prompt the user for input (1 point)
2. Calculating and stating the number of cookies left (1 point)

3. Check whether the input number is between 1 and the number of cookies left (1 point)
4. Print a message when cookies are bought (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

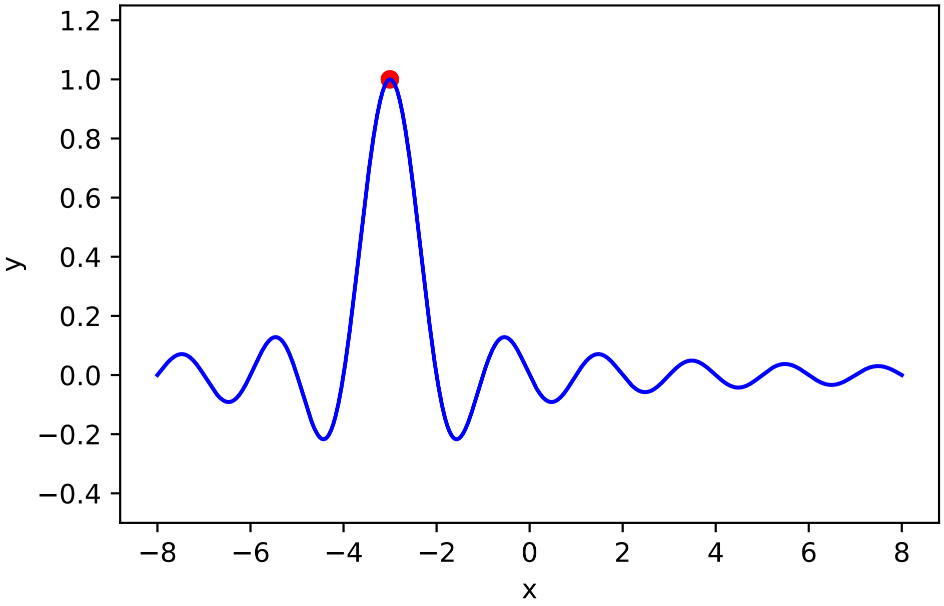
Open the script with name: `script_with_errors.py` and correct the errors. Save the corrected script with file name `solution_3.py`.

The corrected script should plot a sinc curve with equation:

$$y(x) = \text{sinc}(x + 4) = \frac{\sin(x + 4)}{x + 4}$$

Whereby it uses the Numpy `sinc` function. 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:



File of question 3

```
1  # This script contains errors and doesn't run.
2  #
3  # The correct script plots a sinc curve. Afterwards, it saves that
4  # figure to the current folder as a png-file.
5  #
6  # Correct all the errors so that it gives the intended output.
7
8  # Load the necessary libraries
9  import numpy as np
10 import matplotlib.pyplot as plt
11
12 # Initialize the figure
13 fig, ax = subplots()
14
15     # Define the sinc curve
16     a = 3
17     x = np.linspace(-8, 8, 1000)
18     y = np.sinc(x + a)
19
20 # plot the sinc curve and indicate its maximum
21 plot(x, y, color="blue")
22 y_max_index = np.argmax(y)
23 x_max = x[y_max_index]
24 y_max = y(y_max_index)
25 ax.scatter(x_max, y_max, color=red)
26
27 # Set the axes labels and the limits
28 ax.set_xlabel("x")
29 ax.set_ylabel(y)
30  ax.set_ylim([-0.5, 1.25])
31
32 # Save the resulting plot
33 fig.savefig("output_script_with_errors.png")
```

Solution 3

Procedure to reach to the solution:

- On line 13: the `subplots()` function is part of the `matplotlib.pyplot` submodule: replacing the line by `fig, ax = plt.subplots()` will fix the issue.
- On line 15-18: Indentation should be removed.
- On line 21: The `plot(...)` method should be called on the axis object `ax`.

- On line 24: The round brackets should be replaced by square brackets.
- On line 25: The color argument should be replaced by `color="red"`.
- On line 29: The label parameter should be a string: `ax.set_xlabel(y)` should be replaced by `ax.set_ylabel("y")`.
- On line 30: Indentation should be removed.

```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()

# Define the sinc curve
a = 3
x = np.linspace(-8, 8, 1000)
y = np.sinc(x + a)

# plot the sinc curve and indicate its maximum
ax.plot(x, y, color="blue")
y_max_index = np.argmax(y)
x_max = x[y_max_index]
y_max = y[y_max_index]
ax.scatter(x_max, y_max, color="red")

# Set the axes labels and the limits
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_ylim([-0.5, 1.25])

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

Score calculation

7 points to be obtained:

1. Define a figure/axis to plot in (1 point)
2. Having the x and y coordinates (1 point)
3. Plotting the `sinc` curve in correct range (1 point)

4. Calculation of the position of the maximum (1 point)
5. Adding the maximum as a point on the curve (1 point)
6. Saving the output to a file (1 point)
7. Script runs without or with only trivial errors (1 point)

Question 4: 2D density plot

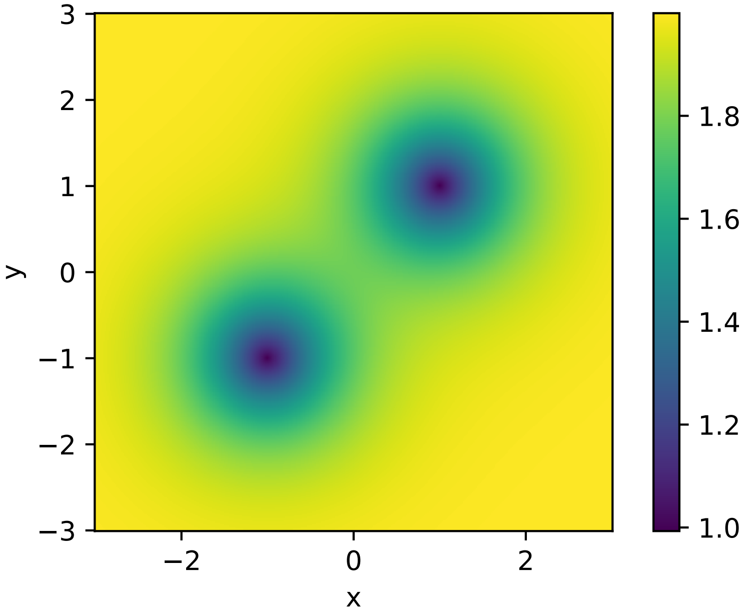
Plot $z(x,y)$, the superposition of two functions (forming a double well) $z(x,y) = z_1(x,y) + z_2(x,y)$ with equations:

$$\begin{cases} z_1(x,y) = \tanh(r_1) = \tanh\left(\sqrt{(x-1)^2 + (y-1)^2}\right) \\ z_2(x,y) = \tanh(r_2) = \tanh\left(\sqrt{(x+1)^2 + (y+1)^2}\right) \end{cases}$$

whereby you can use the tangent hyperbolic function of Numpy: **`tanh()`**. 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)
fig.colorbar(pc, ax=ax)
```

For each of the functions you can use the same (x,y)-values within an 2D interval/domain with $x \in [-3,3]$ and $y \in [-3,3]$ (make your take a sufficiently high number of x values to obtain a smooth density plot). **Save the plot** under the file name: `output_plot_double_well.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 4

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

# Define a 2D grid of coordinates (2D domain)
x = np.linspace(-3, 3, 400)
y = x
xx, yy = np.meshgrid(x, y)

# Calculate the z value for each (x, y) coordinate
rr1 = np.sqrt((xx - 1) ** 2 + (yy - 1) ** 2)
rr2 = np.sqrt((xx + 1) ** 2 + (yy + 1) ** 2)
z_1 = np.tanh(rr1)
z_2 = np.tanh(rr2)
zz = z_1 + z_2

# Plot the density plot
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)

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

Score calculation

9 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. Having the correct interval (1 point)
4. Having the correct expression for the argument of the `tanh()` function (1 point)
5. Understanding how to apply the `tanh` function and obtaining correct z-values from chosen (x, y)-coordinates (1 point)
6. Having a smooth plot (1 point)
7. Style of the plot looks like the example (1 point)
8. Enabling saving the plot (1 point)
9. 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_area.py` which helps to calculate the areas of a square and a circle. The module should contain two functions:

- `area_square(side_length)` which accepts a single floating point number representing the side length of the square. The function should return the area.
- `area_circle(radius)` which accepts a single floating point number for the radius of the circle. The function should return the area of the circle. Remember that the area A of a circle is given by $A = \pi r^2$

Afterwards, import and use this module in a script (with file name: `solution_5.py`) that prompts a user to give the radius of a circle, then uses the above module to calculate its area, and finally prints the area of the circle. See the following example output of a correct script:

```
Please give the radius of the circle: 4.5
The area of a circle with radius 4.5 is: 63.61725123519331
```

Solution 5

The solution exists out of two files:

- `module_area.py`: the module the functionality to calculate areas.
- `solution_5.py`: the main script in which a user is prompted to give the radius of a circle, and then the module function `area_circle(radius)` is called with that radius as argument. Afterwards, the area of the circle is printed.

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 `modulearea.py`

```python
import math

def area_square(side_length):
  return side_length ** 2

def area_circle(radius):
  return math.pi * radius ** 2
```

Listing 2 `solution5.py`

```python
import modulearea as ma

radius = float(input("Please give the radius of the circle: "))
area = ma.circle_area(radius)
print(f"The area of a circle with radius {radius} is: {area}")
```

Score calculation

9 points to be obtained:

1. Creation of a separate module file (1 point)
2. Knowing how to calculate the areas (1 point)
3. Function definitions implementing the area calculation for both circles and squares (1 point)
4. Importing the module (1 point)
5. Prompting the user for input (1 point)
6. Converting the input string to a float (1 point)

7. Calling the `circle_area()` function to calculate the area (1 point)
8. Enabling similar output to the example (1 point)
9. Script runs without or with only trivial errors (1 point)

Question 6: Dictionaries

Consider the following Python list where each item is a dictionary containing some statistical information about three football players: their name, the number of games they played, and the total amount of goals scored by them. Add this list to your script.

```
player_list = [
  {"name": "Mehmet", "matches": 4, "goals": 12},
  {"name": "Ali", "matches": 5, "goals": 23},
  {"name": "Meryem", "matches": 3, "goals": 18}
]
```

Then make the script print out all the player names with their average scored goals per game (do this in an automated manner, by using a loop to iterate over all players in the list). To compute the average you divide their number of goals by the games they played. The output of a correct script should be the following (for the given player information):

```
Mehmet has an average of 3.0 goals/game
Ali has an average of 4.6 goals/game
Meryem has an average of 6.0 goals/game
```

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

Solution 6

```
player_list = [
  {"name": "Mehmet", "matches": 4, "goals": 12},
  {"name": "Ali", "matches": 5, "goals": 23},
  {"name": "Meryem", "matches": 3, "goals": 18}
]

for player in player_list:
  avg = player["goals"]/player["matches"]
  print(f'{player["name"]} has an average of {avg} goals/game')
```

```
Mehmet has an average of 3.0 goals/game
Ali has an average of 4.6 goals/game
Meryem has an average of 6.0 goals/game
```

Score calculation

8 points to be obtained:

1. Understanding how a dictionary is written (1 point)
2. Correctly have a list of dictionaries similar to the one of the task description (1 point)
3. Using a loop to iterate over all players in the list (1 point)
4. Knowing how to index into the dictionary
5. Knowing how to calculate the mean (1 point)
6. Printing the mean for every player (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 `Hotel` which has four attributes

- `hotel_name`: the name of the hotel,
- `n_rooms`: the number of guest rooms (all single rooms),
- `n_guests`: the current number of guests (an integer between 0 and the number of rooms inclusive).
- `guests`: a list containing the names of the guests.

where the first guest gets room number 1, the second guest gets room number 2, etc.

Give the class a constructor that accepts two arguments (in addition to the mandatory `self` argument): the hotel name and the number of rooms of the hotel. Then add two methods:

- `book_room(self, name)`: which books a room under the guest's name.
- `print_rooms(self)`: which prints an overview of the rooms and their guests.

Then use the class to make two hotel objects, one for a hotel at the town-square called "Plaza Hotel" and one for the hotel in the nearby forest called "Forest Hotel". The former hotel has 5 rooms and the latter only 2 rooms. Then try to book guests in the two hotels. Do the above using the following code:

```python
# Add here your class definition
#  ...

# Create the Hotel objects
hotel_plaza = Hotel("Plaza Hotel", 5)
hotel_forest = Hotel("Forest Hotel", 2)

# Bookings at Plaza Hotel
hotel_plaza.book_room("Yeliz")
hotel_plaza.book_room("Shana")
# Print the summary of Plaza Hotel
hotel_plaza.print_rooms()

# Bookings at Hotel Forest
hotel_forest.book_room("Rick")
hotel_forest.book_room("Mortimer")
hotel_forest.book_room("Bob")
```

For a correct script this should give you the following output:

```
Welcome to Plaza Hotel Yeliz, your room No is 1
Welcome to Plaza Hotel Shana, your room No is 2
--------------------------------------------------------------------------------
SUMMARY OF PLAZA HOTEL
Rooms booked: 2 / 5
The current hotel guests are:
Room 1: Yeliz
Room 2: Shana
--------------------------------------------------------------------------------

Welcome to Forest Hotel Rick, your room No is 1
Welcome to Forest Hotel Mortimer, your room No is 2
Sorry Bob, there is no available room at the moment.
```

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

Solution 7

```
class Hotel():

  def __init__(self, hotel_name, n_rooms):
    self.hotel_name = hotel_name
    self.n_rooms = n_rooms
    self.n_guests = 0
    self.guests = []

  def book_room(self, name):
    if self.n_guests < self.n_rooms:
      self.n_guests += 1
      self.guests.append(name)
      print(f"Welcome to {self.hotel_name} {name}, your room No is {len(self.guests)}")
    else:
      print(f"Sorry {name}, there is no available room at the moment.")

  def print_rooms(self):
    print("-" * 80)
    print(f"Summary of {self.hotel_name}".upper())
    print(f"Rooms booked: {self.n_guests} / {self.n_rooms}")
    print("The current hotel guests are: ")
    for i, g in enumerate(self.guests):
      print(f"Room {i+1}: {g}")
    print("-" * 80 + "\n")

# Create the Hotel objects
hotel_plaza = Hotel("Plaza Hotel", 5)
hotel_forest = Hotel("Forest Hotel", 2)

# Bookings at Plaza Hotel
hotel_plaza.book_room("Yeliz")
hotel_plaza.book_room("Shana")
# Print the summary of Plaza Hotel
hotel_plaza.print_rooms()

# Bookings at Hotel Forest
hotel_forest.book_room("Rick")
hotel_forest.book_room("Mortimer")
hotel_forest.book_room("Bob")
```

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 hotel a name and number of rooms (1 point)
4. Having a constructor which allows giving the hotel a name and number of rooms (1 point)
5. Having the method `book_room(self, name)` which adds a guest (1 point)
6. The `book_room(self, name)` method is correctly implemented (1 point)
7. Having the method `print_rooms(self)` which gives a summary of the current guests (1 point)
8. The `print_rooms(self)` method is correctly implemented (1 point)
9. Knowing how to create hotel objects as in the code provided in the question description (1 point)
10. Knowing how to apply object methods as in the code provided in the question description (1 point)
11. Enabling similar output to the example (1 point)
12. Script runs without or with only trivial errors (1 point)