PHOT 110: Introduction to programming

Midterm exam questions & solutions (version A)

Michaël Barbier, Spring semester (2024-2025)

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 first normalized to $20 = 100/5$ (where we round up to the next integer). We then add these for the 5 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:

$$S = \sum_{i=1}^{5} [20 \times m_i/M_i]$$

In the score calculation of the solutions to the questions, we appoint different amount of points. However, every question has the same weight (20/100), 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: Print numbers series

Write a script that prints the series $(n-1)/n$ for first positive numbers $n = 1, \ldots, N$, that is, for a given N (you can take N as a parameter of your script) prints the numbers:

$$0/1,\ 1/2,\ 2/3,\ 3/4,\ \ldots\ ,(N-1)/N$$

where each number is printed on a separate line. If you set `N = 6`, then the output should look similar to:

```
0.0
0.5
0.6666666666666666
0.75
0.8
0.8333333333333334
```

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

Q1: Solution

```python
N = 6
for n in range(1, N+1):
  print((n-1)/n)
```

Q1: Score calculation

6 points to be obtained:

1. Using the correct expression to calculate the values (1 point)
2. Apply the expression on multiple numbers (1 point)
3. Having the correct range of starting numbers: $1, .., N$ (1 point)
4. Printing the resulting values to the console (1 point)
5. Using a loop structure to print one value per line (1 point)
6. Script runs without or with only trivial errors (1 point)

Question 2: Count number of PNG and JPEG files

Define a list with (image) file names as follows:

```python
file_names = ["eye.eps", "garden.png", "horse.jpg", "ball.png", "arrow.pdf", "apple.svg"]
```

Implement a script that counts the number PNG-files (".png") and JPEG-files (".jpg") by verifying the extension. Use a loop structure to verify each file name of the list. You can use the `endswith()` method to verify whether the email address is valid:

```
# Example showing how to use the "endswith()" method
my_string= "hello.txt"
# The endswith() method returns True if the string ends in a
# specific substring (here "txt"), otherwise it returns False
print(my_string.endswith(".txt"))
print(my_string.endswith(".zip"))
```

```
True
False
```

Save your solution as a script with file name: `solution_2.py`. This is the output of a correct working script:

```
Count of PNG and JPEG files
    Number of PNG-files: 2
    Number of JPEG-files: 1
```

Q2: Solution

```
file_names = ["eye.eps", "garden.png", "horse.jpg", "ball.png", "arrow.pdf", "apple.svg"]

number_of_png = 0
number_of_jpg = 0
for file in file_names:
    if file.endswith(".png"):
        number_of_png += 1
    if file.endswith(".jpg"):
        number_of_jpg += 1

print("Count of PNG and JPEG files")
print(f"    Number of PNG-files: {number_of_png}")
print(f"    Number of JPEG-files: {number_of_jpg}")
```

Q2: Score calculation

5 points to be obtained:

1. Correct verification of the file extension (1 point)
2. Performing the verification on each file in a loop (1 point)

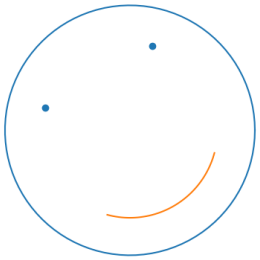
3. (Automatically) Obtaining the correct number of files for each extension (1 point)
4. Having the correct print output (1 point)
5. Script runs without or with only trivial errors (1 point)

Question 3: Correct a Python script

Open the script with name: `script_with_errors.py` and correct the errors.

The correct script prompts the user to type an angle in degrees. It then plots a smiley under that angle and saves it as a PNG-file.

The output plot for 30 degrees is for example:



File of question 3

```
# This script contains errors and doesn't run.
#
# The correct script prompts the user to type an angle in degrees.
# It then plots a smiley under that angle and saves it as a PNG-file.
#
# Correct the errors so that it gives the intended output.

from numpy import sin, cos, pi, linspace, deg2rad
import matplotlib.pyplot as plt

# Prompt the user to input angle in degrees:
angle = input("Please enter an angle in degrees: ")
angle = deg2rad(float(angle)

t_mouth = linspace(0, pi/2, 100) - 3*pi/4
 t = linspace(0, 2*\pi, 100)
```

```
radius_2 = 0.7
radius = 1

# Plot the smiley
fig, ax = plt.subplots()
ax.plot(radius * cos(t), radius * sin(t))
ax.plot(radius_mouth * cos(t_mouth + angle), radius_mouth * sin(t_mouth + angle))
x_eyes = [radius_2 * cos(pi/4 + angle), radius_2 * cosine(3*pi/4 + angle)]
    y_eyes = [radius_2 * sin(pi/4 + angle); radius_2 * sin(3*pi/4 + angle)]
ax.scatter(x_eyes, y_eyes)
ax.set_aspect("equal")
ax.set_axis_off()
plt.show()

fig.savefig("output_script_solution_3.png")
```

Q3: Solution

Procedure to reach to the solution:

- On line 13: Add the missing parentheses at the end of the line.
- On line 16: Remove indentation/space at the beginning of the line.
- On line 16: Remove the backslash in `\pi` to get `pi`.
- On line 25: Remove the indentation/space at the beginning of the line.
- On line 25: Change the separation with a semicolon (`;`) into a komma (‘,’)

```
# This is the corrected script

from numpy import sin, cos, pi, linspace, deg2rad
import matplotlib.pyplot as plt

# Prompt the user to input angle in degrees:
angle = input("Please enter an angle in degrees: ")
angle = deg2rad(float(angle))

t_mouth = linspace(0, pi/2, 100) - 3*pi/4
t = linspace(0, 2*pi, 100)
radius_mouth = 0.7
radius = 1

# Plot the smiley
```

```
fig, ax = plt.subplots()
ax.plot(radius * cos(t), radius * sin(t))
ax.plot(radius_mouth * cos(t_mouth + angle), radius_mouth * sin(t_mouth + angle))
x_eyes = [0.7 * cos(pi/4 + angle), 0.7 * cos(3*pi/4 + angle)]
y_eyes = [0.7 * sin(pi/4 + angle), 0.7 * sin(3*pi/4 + angle)]
ax.scatter(x_eyes, y_eyes)
ax.set_aspect("equal")
ax.set_axis_off()
plt.show()

fig.savefig("output_script_solution_3.png")
```

Q3: Score calculation

7 points to be obtained:

1. Allowing for user input in degrees (1 point)
2. Correct conversion of the input (1 point)
3. Having the contour of the smiley (1 point)
4. Plotting the facial features of correct size (1 point)
5. Plotting the smiley under the correct angle (1 point)
6. Saving the output to a file (1 point)
7. Script runs without or with only trivial errors (1 point)

Question 4: Repeated input

Repeatingly prompt the user to input an integer number between 1 and 100 until the provided integer is divisible by 9. Verify whether the number is within the given range. If the number is divisible by 9, then print a message telling how many times 9 fits in the number (see example), otherwise let the user try again. Allow the user also to try again when: the user types a number which is too small, too large, or types invalid input.

Example:

```
Give an integer divisible by 9 number in interval [1, 100]: 125
The number 125 is invalid, please try again.

Give an integer divisible by 9 number in interval [1, 100]: eighteen
The number eighteen is invalid, please try again.

Give an integer divisible by 9 number in interval [1, 100]: 18
```

```
The number you provided is divisible by 9 and:
  18 = 2 times 9
```

Save your solution as a script with file name: `solution_4.py`.

Q4: Solution

```python
while True:
    n_str = input("Give an integer divisible by 9 in interval [1, 100]: ")
    if n_str.isdigit():
        n = int(n_str)
        if (n <= 100) and (n >= 1):
            if n % 9 == 0:
                break
    else:
        pass
    print(f"The number {n_str} is invalid, please try again.")

print("The number you provided is divisible by 9 and:")
print(f"  {n} = {int(n/9)} times 9")
```

Q4: Score calculation

6 points to be obtained:

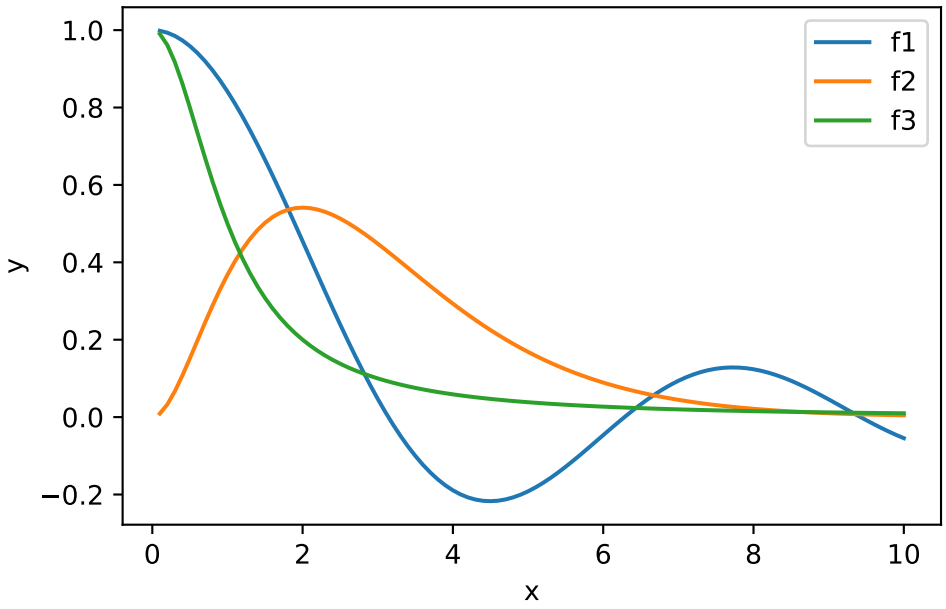
1. Prompt the user for input (1 point)
2. Convert to an integer (1 point)
3. Verify whether the number is divisible by 9 (1 point)
4. Verify whether the number is within the correct interval (1 point)
5. Check the validity of the number (1 extra point, **we didn't see this in the theory yet**)
6. Prompt the user again if input is not appropriate (1 point)
7. Script runs without or with only trivial errors (1 point)

Question 5: Plot functions

Plot a graph with the following functions (in interval $[0.1, 10]$):

$$f_1(x) = \frac{\sin(x)}{x}$$
$$f_2(x) = x^2 \exp(-x)$$
$$f_3(x) = \frac{1}{1+x^2}$$

You can make use of the functions in the `numpy` library: `exp()`, `sin()`, and the value of `pi`. Take a sufficiently high number of x values so the curve looks smooth, you can use the `numpy` function `linspace()`. **Save the plot** under the file name: `output_plot_math.png`. **Save your solution** as a script with file name: `solution_5.py`.



Q5: Solution

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

x = np.linspace(0.1, 10, 100)
f1 = np.sin(x) / x
f2 = x**2 * np.exp(-x)
f3 = 1 / (x**2 + 1)

fig, ax = plt.subplots()
```

```
ax.plot(x, f1)
ax.plot(x, f2)
ax.plot(x, f3)
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.legend([r"f1", r"f2", r"f3"])
plt.show()
```

Q5: Score calculation

9 points to be obtained:

1. Obtaining the correct expressions for the three functions (1 point)
2. Creating multiple x-values within the interval (1 point)
3. Having the correct interval (1 point)
4. Obtaining the correct y-values from chosen x-values for at least one of the curves (1 point)
5. Having all curves (1 point)
6. Having smooth plots (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)