PHOT 110: Introduction to programming

Midterm exam questions & solutions, version C

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 $\sqrt{n(n+3)}$ for numbers $n = 1, \ldots, N$, that is, for a given N (you can take N as a parameter of your script) prints the numbers:

$$1,\ \sqrt{4},\ \sqrt{10},\ \sqrt{18},\ \ldots\ ,\ \sqrt{N(N+3)}$$

where each number is printed on a separate line. If you set `N = 5`, then the output should look similar to:

```
2.0
3.1622776601683795
4.242640687119285
5.291502622129181
6.324555320336759
```

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

Q1: Solution

```
from math import sqrt
N = 5
for n in range(1, N+1):
  print(sqrt(n*(n+3)))
```

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:

Create a script that prompts the user to input a sentence with a minimum of 5 words. If the sentence contains less than 5 words, allow the user to try again until he/she provides a long enough sentence. You can check the number of words e.g. by counting spaces.

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

```
Please provide a sentence (min. 5 words): It is sunny weather.
This sentence has not enough words, please try again.
Please provide a sentence (min. 5 words): The apples.
This sentence has not enough words, please try again.
```

```
Please provide a sentence (min. 5 words): This book is very nice.
Thank you for your input.
```

Q2: Solution

```python
min_words = 5
while True:
    sentence = input("Please provide a sentence (min. 5 words): ")
    if len(sentence.split(" ")) >= min_words:
        break
    else:
        print("This sentence has not enough words, please try again.")

print("Thank you for your input.")
```

Q2: Score calculation

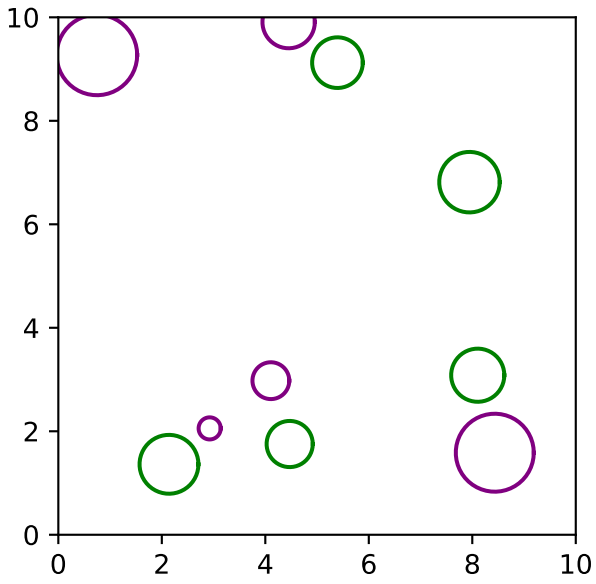
5 points to be obtained:

1. Prompting the user for input (1 point)
2. Correct extraction of the number of words in the sentence (1 point)
3. Repeatedly allowing the user to provide a sentence until it is long enough (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_c.py` and correct the errors.

The correct script plots a series of circles at random positions and with random radii. It then saves the plot as a PNG-file under the name: `output_script_with_errors_c.png`.



File of question 3

```
# This script contains errors and doesn't run.
# Correct the errors so that it gives the intended output.

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

fig, ax = plt.subplots()
t = numpylinspace(0, 2 * pi)
colors = "green", "purple"]

for i in range(10):
    x0 = random.uniform(0, 10)
    y0 = random.uniform(0, 10)
    r = random.uniform(0.2, 1)
    xx = x0 + rr * cos(t)
    yy = y0 + r * sin(t)
     ax.plot(xx, yy, color=colors[i % 2])

ax.set_aspect("equal")
ax.set_xlim([0, "10"])
```

```
ax set_ylim([0, 10])
fig.savefig("output_script_with_errors_c.png")
```

Q3: Solution

Procedure to reach to the solution:

- On line 9: Change `numpylinspace` to `linspace`.
- On line 10: Add left `[` square bracket before `"green"`.
- On line 16: Change `rr` to `r`.
- On line 18: Remove the indentation/space at the beginning of the line.
- On line 21: Change string `"10"` into integer `10`
- On line 22: Add the missing dot, i.e. change `ax set_ylim` to `ax.set_ylim`

```
# This is the corrected script


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

fig, ax = plt.subplots()
t = linspace(0, 2*pi)
colors = ["green", "purple"]

for i in range(10):
    x0 = random.uniform(0, 10)
    y0 = random.uniform(0, 10)
    r = random.uniform(0.2, 1)
    xx = x0 + r*cos(t)
    yy = y0 + r*sin(t)
    ax.plot(xx, yy, color=colors[i % 2])

ax.set_aspect("equal")
ax.set_xlim([0, 10])
ax.set_ylim([0, 10])

plt.savefig("output_script_with_errors_c.png")
```

Q3: Score calculation

7 points to be obtained:

1. Having a parameter `t` with sufficient points (1 point)
2. Creating the correct number of random (x, y)-coordinates for the circles (1 point)
3. Creating the correct number of random radii for the circles (1 point)
4. Being able to plot circles (1 point)
5. Plotting the circles with correct colors (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

Prompt the user to input an integer number N between 100 and 200. Repeatedly divide the number by 2 or if not possible by 3 and print the result. If it cannot divided by 2 and cannot divided by 3 stop the program and print the resulting number.

Example:

```
Give an integer in interval [100, 200]: 120
120 / 2 = 60
60 / 2 = 30
30 / 2 = 15
15 / 3 = 5
The result is: 5
```

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

Q4: Solution

```python
n_str = input("Give an integer in interval [100, 200]: ")
n = int(n_str)

while True:
    if n % 2 == 0:
        print(f"{n} / 2 = {n / 2}")
        n = n / 2
    elif n % 3 == 0:
        print(f"{n} / 3 = {n / 3}")
        n = n / 3
```

```
    else:
        break
print(f"The result is: {n}")
```

Q4: Score calculation

6 points to be obtained:

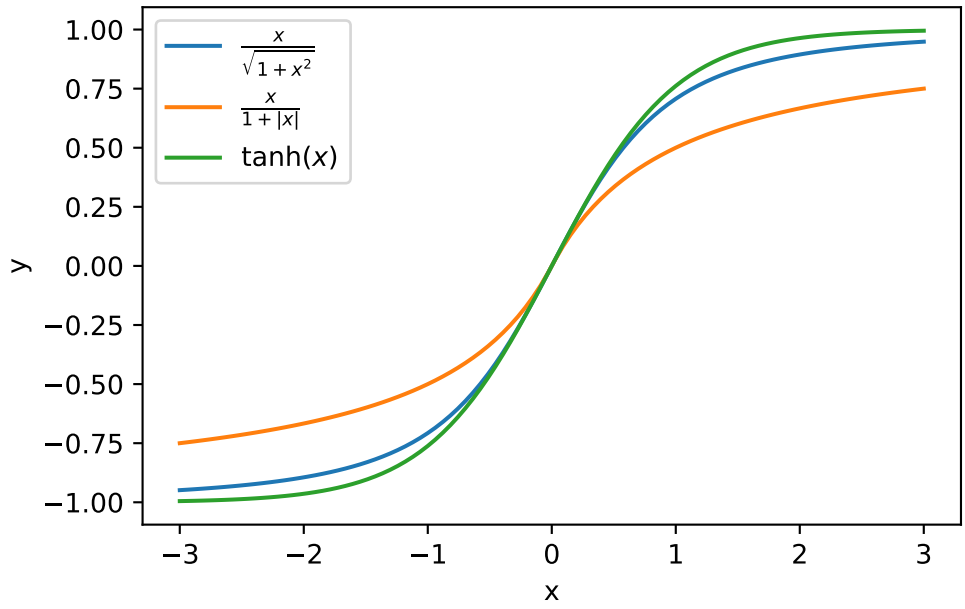
1. Prompt the user for input (1 point)
2. Convert to an integer (1 point)
3. Verify whether the number is divisible by 2 or 3 (1 point)
4. Print the division in correct output format (1 point)
5. Check the validity of the number or range of the number (for an extra point)
6. Repeat the division in a correctly working for loop (1 point)
7. Script runs without or with only trivial errors (1 point)

Question 5: Plot functions

Plot a graph with the following sigmoid functions (in interval $[-3, 3]$):

$$f_1(x) = \frac{x}{\sqrt{1+x^2}}$$
$$f_2(x) = \frac{x}{1+|x|}$$
$$f_3(x) = \tanh(x)$$

You can make use of the functions in the `numpy` library: `sqrt()`, `tanh()`, `abs()`. 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
from numpy import sqrt, abs, tanh
import matplotlib.pyplot as plt

x = np.linspace(-3, 3, 100)
f1 = x / sqrt(1 + x**2)
f2 = x / (1 + abs(x))
f3 = tanh(x)

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"$\frac{x}{\sqrt{1+x^2}}$", r"$\frac{x}{1 + |x|}$", r"$\tanh(x)$"])

plt.savefig("output_plot_math.png")
```

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)