PHOT 110: Introduction to programming

Midterm 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. The points of each question are first normalized to 20 = 100/5 (where we round up to the first 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:

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

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 1A: Print the third powers

Write a script that prints the first N positive integer numbers to the third power, that is, for a given N (you can take N as a parameter of your script) prints the numbers:

$$1^3,\ 2^3,\ 3^3,\ 4^3,\ \dots,\ N^3$$

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

```
1
8
27
64
125
```

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

Solution 1A

```python
N = 5
for n in range(1, N+1):
  print(n ** 3)
```

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: Plot the error function

Implement a script that plots the error function:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\,\mathrm{d}t$$

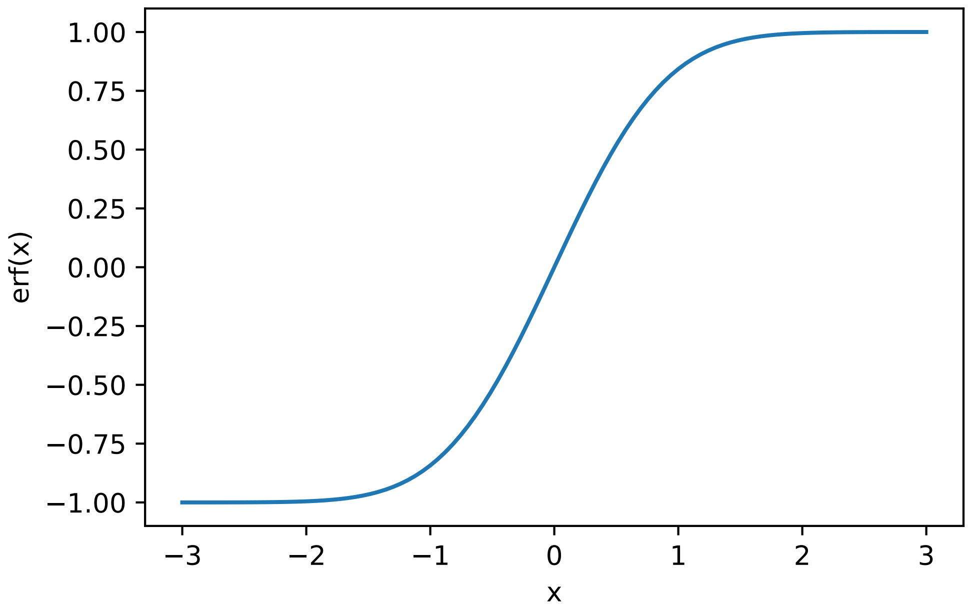
You can make use of the error function in the **`math`** library: **`math.erf()`** which takes one argument (the x-value in the above formula) and can be used as in the following example:

```python
import math

# This example prints the error function for x = 0.8
y = math.erf(0.8)
print(y)
```

```
0.7421009647076605
```

Calculate the error function for multiple x-values in interval $[-3, 3]$, and make a line plot (take a sufficiently high number of x values so the curve looks smooth). **Save the plot** under the file name: `output_plot_math_erf.png`. Save your solution as a script with file name: `solution_2.py`.



Solution 2A

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

x = np.linspace(-3, 3, 100)
y = np.zeros_like(x)
for i in range(len(x)):
  y[i] = math.erf(x[i])

fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_xlabel("x")
ax.set_ylabel("erf(x)")
fig.savefig("output_plot_math_erf.png")
```

Score calculation

6 points to be obtained:

1. Knowing how to apply the correct function (1 point)
2. Creating multiple x-values within the interval (1 point)
3. Obtaining the correct y-values from chosen x-values (1 point)
4. Having a smooth plot (1 point)
5. Enabling saving the plot (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.

The corrected script prompts the user to type a sentence. It then reverses the order of the words, prints this reversed sentence on the console and saves it to a text-file.

For example when you would enter: `This is my sentence.` as input, then the script will print the following output to the console:

```
sentence. my is This
```

It also saves the same sentence to a text file with file name: `output_text_script_with_errors.txt`

File of question 3A

```python
# This script contains errors and doesn't run.
#
# The correct script prompts the user to type a sentence.
# It then reverses the order of the words, prints this
# reversed sentence on the console and saves it to a
# text-file.
#
# Correct the errors so that it gives the intended output.

# Prompt the user to input a sentence:
sentence = input("Please enter a sentence: ")
```

```
12
13 # Split the sentence into words and reverse the order
14 sentence.split() = word_list
15 word_list_reversed =
16     reversed(word_list)
17 # Put the sentence back together using the reversed ordered words.
18 sentence_reversed = ' '.join(word_list_reversed)
19
20 # Print the reversed sentence
21  print("The reversed version of the sentence is:\n" + reversed)
22 # Save the sentence to a file
23 f = open(file="output_text_script_with_errors.txt", mode="w+"):
24     f.write(sentence_reversed)
25 f.close()
```

Solution 3A

Procedure to reach to the solution:

- On line 14: variable `word_list` should be on the left side, and `sentence.split()` on the right side.
- On line 15: `reversed(word_list)` should not start a new line.
- On line 21: indentation/space before `print` should be removed.
- On line 21: the variable “sentence_reversed” should be printed instead of “reversed”
- On line 23: colon symbol “:” at end of statement needs to be removed.
- On line 24: indentation/space before `f.write(sentence_reversed)` should be removed.

```
# This is the corrected script.

# Prompt the user to input a sentence:
sentence = input("Please enter a sentence: ")

# Split the sentence into words and reverse the order
word_list = sentence.split()
word_list_reversed = reversed(word_list)
# Put the sentence back together using the reversed ordered words.
sentence_reversed = ' '.join(word_list_reversed)

# Print the reversed sentence
print("The reversed version of the sentence is:\n" + sentence_reversed)
# Save the sentence to a file
f = open(file="output_text_script_with_errors.txt", mode="w+")
```

```
f.write(sentence_reversed)
f.close()
```

Score calculation

7 points to be obtained:

1. Enabling the splitting of the sentence in words (1 point)
2. Reversing the order of the words (1 point)
3. Having the reversed list of words in a variable (1 point)
4. Being able to print the adapted list or sentence (1 point)
5. Having the correct print output (1 point)
6. Saving the output to a text file (1 point)
7. Script runs without or with only trivial errors (1 point)

Question 4A: User input and error handling

Prompt the user to input a floating point number between 0 and 20. print the integer part and the rest part. Print also whether the number is larger than or equal to 10. Extend the program by catching any errors (using the `try` and `except` keywords) when the user types a number which is too small, too large, or types invalid input. Allow the user to try again in that case.

Example:

```
Give a decimal number in interval [0, 20]: 25
The number 25 is invalid, please try again.

Give a decimal number in interval [0, 20]: twenty
The number twenty is invalid, please try again.

Give a decimal number in interval [0, 20]: 9.4
The number you provided has:
  integer part: 9
  decimal part: 0.4
The number is smaller than 10
```

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

Solution 4A

```python
import math

while True:
    number_str = input("Give a decimal number in interval [0, 20]: ")
    try:
        number = float(number_str)
        if (number <= 20) and (number >= 0):
            print("The number you provided has:")
            print(f"\tinteger part: {math.floor(number)}")
            print(f"\tdecimal part: {number - math.floor(number)}")
            if number < 10:
                print("The number is smaller than 10")
            else:
                print("The number is larger than or equal to 10")
            break
        else:
            print("The number " + number_str + " is out of range, please try again.")
    except ValueError:
        print("The number " + number_str + " is invalid, please try again.")
```

Score calculation

9 points to be obtained:

1. Prompt the user for input (1 point)
2. Convert to a float if required (1 point)
3. Calculate the integer and the decimal part (1 point)
4. Print the integer and the decimal part (1 point)
5. Verify whether the number is larger than or equal to 10 (1 point)
6. Verify whether the number is within the correct interval (1 point)
7. Check the validity of the number (1 point)
8. Prompt the user again if input is not appropriate (1 point)
9. Script runs without or with only trivial errors (1 point)

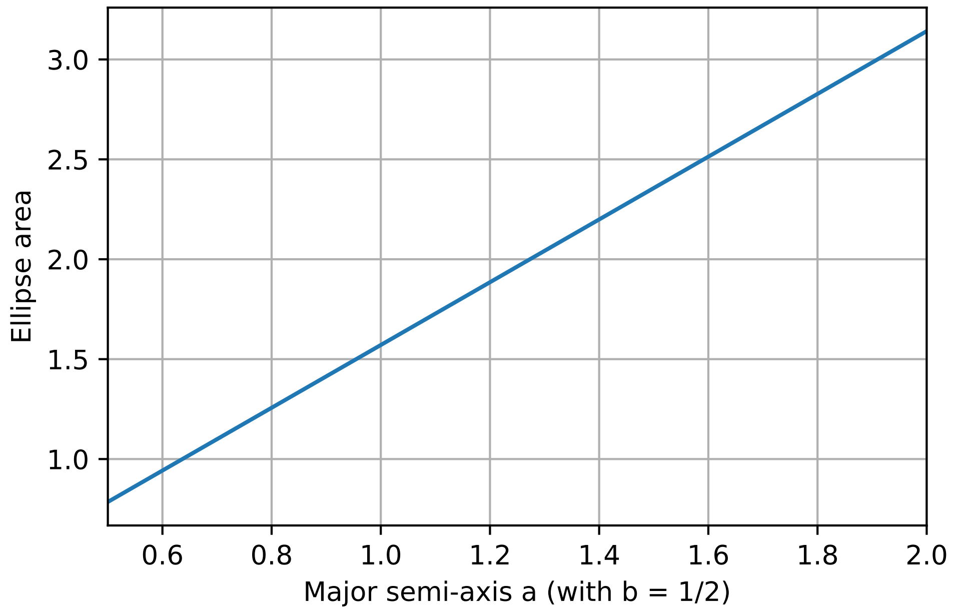
Question 5: Creating and using modules

Create a **module** (a separate script file) with file name `module_ellipse.py` which contains two functions: `perimeter_ellipse(a, b)` and `area_ellipse(a, b)` which calculate the perimeter and the area of an ellipse. The parameters `a` and `b` are the major and minor semi-axis.

Use the following formulas to calculate the (approximate) perimeter P and the area A of the ellipse:

$$A = \pi ab$$
$$P = \pi \left[3\,(a+b) - \sqrt{(3a+b)(a+3b)} \right]$$

Afterwards, import and use this module in a script (with file name: `solution_5.py`) to plot the **area** as function of a where you fix the minor semi-axis $b = 1/2$. Take the interval for parameter $a \in [0.5, 2]$. Use a line plot with enough points to get a smooth curve. Let the script **save the plot** to a file named `output_plot_ellipse.png`. The saved file should contain a graph similar to the one below:



Solution 5A

The solution exists out of two files:

- `module_ellipse.py`: the module to calculate the ellipse properties
- `solution_5.py`: the main script in which the area of an ellipse is plotted.

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 `moduleellipse.py`

```
import numpy as np

def area_ellipse(a, b):
  area = np.pi * a * b
  return area

def perimeter_ellipse(a, b):
  perimeter = np.pi * ( 3*(a+b) - np.sqrt((3*a + b)*(a + 3*b)) )
  return perimeter
```

Score calculation

10 points to be obtained:

1. Creation of a separate module file (1 point)
2. Using the correct expressions for area and perimeter (1 points)
3. Function definition for the area and perimeter calculation (1 points)
4. Importing the module (1 point)
5. Creating a list or vector of x-values, representing the size a of the major semi-axis (1 point)
6. Calling the area function with the correct parameters (1 point)
7. Having a correct plot of area as function of major semi-axis a (1 point)
8. Having a plot looking similar to the one shown in the question (1 point)
9. Enabled saving of the plot (1 point)
10. Script runs without or with only trivial errors (1 point)

Listing 2 `solution5.py`

```
import matplotlib.pyplot as plt
import moduleellipse as me

b = 1/2
a = np.linspace(0.5, 2, 100)
y = me.area_ellipse(a, b)

fig, ax = plt.subplots()
ax.plot(a, y)
ax.set_xlabel("Major semi-axis a (with b = 1/2)")
ax.set_ylabel("Ellipse area")
ax.set_xlim([0.5,2])
ax.grid()
fig.saveas("output_plot_ellipse.png")
```