PE1: PE Practice

Master in Informatics and Computing Engineering
Programming Fundamentals
Instance: 2018/2019

0. Introduction

Some important information about this PE (Practical on computer evaluation):

- The password to enter the test is **`EnterNow`**.
- You have **75 minutes** to answer the 5 questions of the test
- **No collaboration** between students is allowed
- The presence on the table and the use of mobile phones or any other electronic devices **is forbidden**
- The Python code that answers each question is saved in a different file, with the name required in the question
- **Before the time expires** you must *upload* a zip (**`pe1.zip`**) with the Python code of all your answers collected in a folder named **`PE1`**; as you have 2 attempts, you should try the submission procedure 10 minutes before the time expires
- ~~Download the PDF file **`PE1: Assignment (Practice)`** with the questions and~~ start answering using Spyder3
- Your are allowed to use the Book provided in PDF as **`Consultation Book`**
- The forum will not be available, as well as the submissions to RE Weekly away assignments RE01-RE04

1. Interests

The formula for computing the final amount if one is earning compound interest is given on Wikipedia as:

$$A = P\left(1 + \frac{r}{n}\right)^{nt}$$

Where:

- P = principal amount (the amount that the interest is provided on)
- r = the interest rate
- n = the frequency that the interest is paid out (per year)
- t = the number of years that the interest is calculated for

Use Spyder3 to create a new file named **`question1.py`** in your folder named **`PE1`**.

In the file, develop and test a Python program that compares the final amounts (A) for two different interest rates (r) and two different payment frequencies (n), given by user input. Consider the principal amount of 1000 (P=1000) and one year (t=1).

For example, for the given values, the output of the program is:

```
For r=0.05 and n=2 you'll have 1050.625
For r=0.06 and n=1 you'll have 1060.0
```

Do not forget to save your program in the file **`question1.py`**.

2. Divisors

Write a Python program that given an integer **num**, provided by the user, prints the sum of all its divisors. A divisor of an integer number n is a number which divides n without remainder.

For example:

- for num=35 the output is **48** (1+5+7+35)
- for num=27 the output is **40** (1+3+9+27)
- for num=23 the output is **24** (1+23)

Save your program in the file **question2.py** inside the folder **PE1**.

3. Babylonian square-root

Write a Python program that uses the Babylonian method for printing square roots.

For a number **num** given by user input, to find its square root, do the following:

1. Make an initial guess: any positive number x_0<**num**
2. Improve the guess: apply the formula $x_1=(x_0+num/x_0)/2$; the number x_1 is a better approximation to **sqrt(num)**
3. Iterate until convergence: apply the formula $x_{n+1}=(x_n+num/x_n)/2$ until the process converges; convergence is achieved when the digits of x_{n+1} and x_n agree on 2 decimal places

For example:

- for num=20 the output is **4.472**
- for num=25 the output is **5.000**

Save your program in the file **question3.py** inside the folder **PE0**.

4. Grading FPRO

Write a Python program that, given the four components of the FPRO grade, by user input, returns the student's grade, an integer from 0 to 20, by using the formula:

grade = (LE + RE + 4 * PE + 4 * TE) / 50

The program returns:

- "**Input error**", if the any of the components is not between 0 and 100
- "**RFC**", if the PE<40 or the TE<40
- the grade as an integer, otherwise

For example:

- for LE=100, RE=100, PE=100, TE=100, the output is **"20"**
- for LE=200, RE=100, PE=100, TE=100, the output is **"Input error"**
- for LE=45, RE=85, PE=35, TE=100, the output is **"RFC"**

Save your program in the file **question4.py** inside the folder **PE1**.

5. Prime numbers

Write a Python program that outputs all the prime numbers within an interval of numbers between **`lower`** and **`upper`**, given by user input.

For example:

- for lower=`2` and upper=`23` the output is: **`Prime numbers between 2 and 23 are: 2 3 5 7 11 13 17 19 23`**
- for lower=`5` and upper=`27` the output is: **`Prime numbers between 5 and 27 are: 5 7 11 13 17 19 23`**
- for lower=`-2` and upper=`5` the output is: **`Prime numbers between -2 and 5 are: 2 3 5`**

Save your program in the file **`question5.py`** inside the folder **`PE1`**.

The end.

FPRO, 2018/19