Programming Assignment 05
-
Functions

Instructions

This programming assignment consists of **2 programming exercises**.
You have to:

1. **create** Python files on your computer (be careful of file names)
2. **write** the program according to the assignment
3. **verify** on your computer that it works (run them and check the output in the shell)
4. **upload** the files to Gradescope (upload directly the **.py** files, **not** a **.zip** file)
5. **check** the autograder report on Gradescope
6. **go back to step 2** if necessary

The autograder will evaluate your code for a few testcases. If some testcases fail, the autograder should show you what is your code output, and what is the expected output.

The autograder will give you a score based on the testcases, but a grader will manually evaluate your coding style after the deadline. Style represents 30% of the coding assignment grade.

Exercise 1 - Reduce a fraction to its lowest terms

A fraction can be reduced to an equivalent fraction with a smaller numerator and a smaller denominator, this is called simplifying or reducing a fraction.

Write the function `greatest_common_divisor` (in the file `exercise1.py`) that does the following.

Function `greatest_common_divisor(n, m)`

- takes 2 parameters `n, m` → type `int`
- returns the greatest common divisor of the two integers → type `int`

Write the function `greatest_common_divisor` (in the file `exercise1.py`) that does the following.

Function `reduce_fraction(n, m)`

- takes 2 parameters `n, m` → type `int` that , respectively, represent the numerator and denominator of a fraction as its only two parameters.
- calls an other function `greatest_common_divisor` and use the returned value to reduce the fraction to its lowest terms.
- returns the numerator and denominator of the reduced function if the function can be reduced or the original numerator and denominator if the function cannot be reduced → type `int`

Then, **write a program** (in the `main()` in the file `exercise1.py`) that does the following.

Program `main()`

- ask the user to enter a numerator and denominator
- **prints out** the reduced function or a message that the function cannot be reduced as shown in the examples below.

Note: You can assume the user enters valid integers.

Sample examples (the user input is in red, the printed output is in blue, and the prompt is in black):

```
Enter the numerator: > 96
Enter the denominator: > 36
The fraction 96/36 can be reduced
to 8/3
```

```
Enter the numerator: > 91
Enter the denominator: > 17
The fraction 91/17 cannot be
reduced
```

Exercise 2 - Perfect Numbers

A proper divisor of a positive integer N, is a positive integer less than N that divides evenly into N leaving no remainder. For example, the proper divisors of 18 are: 1, 2, 3, 6 and 9.

Write the function `proper_divisors` (in the file `exercise2.py`) that does the following:

Function `proper_divisors(N)`

- takes 1 parameter `N` → type `int`
- returns the sum of the proper divisors of N → type `int`

An integer, N, is said to be perfect when the sum of all of the proper divisors of N is equal to N. For example, 28 is a perfect number because its proper divisors are 1, 2, 4, 7 and 14, and 1 + 2 + 4 + 7 + 14 = 28.

Write the function `perfect_number` (in the file `exercise2.py`) that does the following:

Function `perfect_number(N)`

- takes 1 parameter `N` → type `int`
- calls the function `proper_divisor`
- returns whether the number is a perfect number → type `bool`

Then, **write a program** (in the `main()` in the file `exercise2.py`) that does the following.

Program `main()`

- calls the function `perfect_number` for numbers between 1 and 10,000
- **prints out** the number if it is a perfect number