Python essentials

Ex01

Write a loop that prints all the numbers from 1 to 10.

Ex02

Write a loop that prints all the numbers from 1 to 100.

Ex03

Write a loop that prints all the numbers from 1 to 100, but only the multiples of 3.

The output should look like this:

```
3
6
9
12
...
96
99
```

Ex04

Write a loop that prints all the numbers from 1 to 100, but only the multiples of 5.

The output should look like this:

```
5
10
15
20
...
```

```
95
100
```

Ex05

Write a loop that prints all the numbers from 1 to 100, but only the multiples of 3 or 5.

The output should look like this:

```
3
5
6
9
10
12
...
95
96
99
100
```

Ex06

Write a loop that sums all the numbers from 1 to 100 and then prints the sum.

The result should be `5050`.

Ex07

Write a loop that sums all the numbers from 1 to 100, but only the multiples of 3 or 5.

The result should be `2418`.

Ex08

Write a loop that sums all the numbers from 1 to 1000, but only the multiples of 3 or 5.

The result should be `234168`.

Ex09

Write a function called `square` that accepts a number and returns its square.

Examples:

```
print(square(2))  # 4
print(square(3))  # 9
```

Ex10

Write a function called `sum_first` that accepts a number and then uses a loop to sum the first numbers up until that value.

Examples:

```
print(sum_first(1))  # 1
print(sum_first(2))  # 3
print(sum_first(3))  # 6
print(sum_first(100))  # 5050
```

Ex11

Write a function called `sum_first_squares` that accepts a number and then uses a loop to sum the first squares up until that value.

For example, `sum_first_squares(4)` should do $1^2 + 2^2 + 3^2 + 4^2 = 30$.

Examples:

```
print(sum_first_squares(1))  # 1
print(sum_first_squares(4))  # 30
```

```
print(sum_first_squares(10))  # 385
print(sum_first_squares(100))  # 338350
```

Ex12

Write a function called `sum_diffs` that accepts a number `n` and then computes the difference between `square(sum_first(n))` and `sum_first_squares(n)`.

Ex13

Write a function `is_divisor(num, div)` that checks if `div` is a divisor of `num`.

Examples:

```
print(is_divisor(5, 2))  # False
print(is_divisor(10, 5))  # True
print(is_divisor(17, 7))  # False
```

Ex14

Write a function `print_divisors(n)` that uses a loop to print all the divisors of the number `n`.

For example, `print_divisors(12)` should produce this output:

```
1
2
3
4
6
12
```

Ex15

Write a function `sum_divisors(n)` that uses a loop to sum all of the divisors of the number `n`.

Examples:

```
print(sum_divisors(1))  # 1
print(sum_divisors(2))  # 3
print(sum_divisors(3))  # 4
print(sum_divisors(12))  # 28
```

Ex16

An "abundant number" is a number for which the sum of its divisors is greater than twice the number.

For example, 12 is abundant because the divisors of 12 are 1, 2, 3, 4, 6, and 12, and they sum up to 28, which is larger than 24 (2×12).

Write a function `is_abundant(n)` that checks whether `n` is abundant or not.

Examples:

```
print(is_abundant(1))  # False
print(is_abundant(3))  # False
print(is_abundant(12))  # True
print(is_abundant(20))  # True
print(is_abundant(21))  # False
```

Ex17

Write a function `collatz_step(n)` that does one of two things:

1. it returns the value $n/2$ if the number `n` is even; and
2. it returns the value $3n + 1$ if the number `n` is odd.

Examples:

```
print(collatz_step(1))  # 4
print(collatz_step(2))  # 1
print(collatz_step(3))  # 10
print(collatz_step(4))  # 2
```

Ex18

Write a function `collatz_sequence(n)` that prints the results of applying the function `collatz_step` repeatedly, until the result is 1.

For example, if you do `collatz_sequence(13)` this is the output:

```
40
20
10
5
16
8
4
2
1
```

Ex19

Write a function `collatz_length(n)` that computes the length of the Collatz sequence of the exercise before.

In other words, `collatz_length(n)` counts how many steps it takes to go from `n` to the number `1` by applying the function `collatz_step` repeatedly.

Examples:

```
print(collatz_length(1))  # 0
print(collatz_length(2))  # 1
print(collatz_length(3))  # 7
```

```
print(collatz_length(4))  # 2
print(collatz_length(13))  # 9
```

Ex20

Write a function `factorial(n)` that implements the factorial function from mathematics.

The factorial, from maths, is this:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$$

Here are some examples:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

$$7! = 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 5040$$

So, the function `factorial` should produce these results:

```
print(factorial(5))  # 120
print(factorial(7))  # 5040
```

Ex21

Write a function called `is_divisible` that accepts a number `n`.

If the number `n` is odd, the function should print the message `"Your number is divisible by 2."`.
If the number
`n` is even, the function should print the message `"Your number is NOT divisible by 2."`.

Here are some examples:

```
is_divisible(2)  # Your number is divisible by 2.
```

```
is_divisible(3)  # Your number is NOT divisible by 2.
```

Important: because the function `is_divisible` already uses `print` inside it, we don't need to put a `print` when we use the function.

Ex22

Write a function called `is_divisible2` that accepts a number `n`.

If the number `n` is odd, the function should print the message `"<n> is divisible by 2."`.
If the number
`n` is even, the function should print the message `"<n> is NOT divisible by 2."`.

The value of your number `n` should be substituted in the message.

Here are some examples:

```
is_divisible(2)  # 2 is divisible by 2.
is_divisible(3)  # 3 is NOT divisible by 2.
```

Important: because the function `is_divisible2` already uses `print` inside it, we don't need to put a `print` when we use the function.

Ex23

Write a function called `is_divisible3` that accepts a number `n`.

If the number `n` is odd, the function should print the message `"The number <n> is divisible by 2."`.
If the number
`n` is even, the function should print the message `"The number <n> is NOT divisible by 2."`.

The value of your number `n` should be substituted in the message.

Here are some examples:

```
is_divisible(2)  # The number 2 is divisible by 2.
is_divisible(3)  # The number 3 is NOT divisible by 2.
```

Important: because the function `is_divisible3` already uses `print` inside it, we don't need to put a `print` when we use the function.

Ex24

Write a function called `rectangle_area` that accepts two numbers `width` and `height` and computes the area of a rectangle with the given width and the given height.

Here are some examples:

```
print(rectangle_area(10, 5))  # 50
print(rectangle_area(5, 10))  # 50
print(rectangle_area(5, 5))  # 25
```

Ex25

Write a function called `greet_person` that accepts two pieces of information: a `name` and an `age`.

The function should then use `print` to produce the following message: `"Hey, my name is <name> and I am <age> years old."`. The values that are passed into the function must be substituted into the message.

Here are some examples:

```
greet_person("John", 27)  # Hey, my name is John and I am 27 yea
greet_person("Anna", 45)  # Hey, my name is Anna and I am 45 yea
```

Important: because the function `greet_person` already uses `print` inside it, we don't need to put a `print` when we use the function.