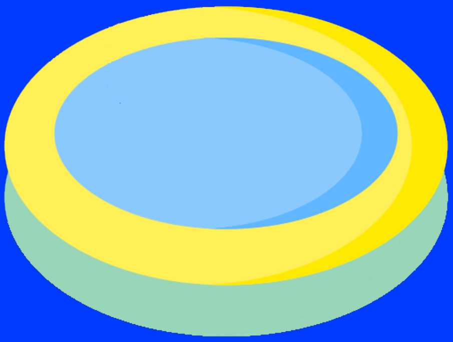
WADING POOL

< 08 - BUILT-IN COMMON FUNCTIONS />



WADING POOL

Code wars
In addition to the tasks below, you must go as far as possible in this code wars collection. Try to solve the first one until the last one without skipping challenges!
Work on it as soon as you have a bit of time, or whenever you need a break in you day!

Built-in functions

Imagine you're building a house, and you decide to carve your own bricks from scratch.

Sounds like a lot of work, right? Why reinvent the wheel?
Why spend hours shaping custom functions when something already got your back!?
Python's built-in functions will make your life easier, like a set of powerful tools.

Task 1.1

Use some built-in functions to get:

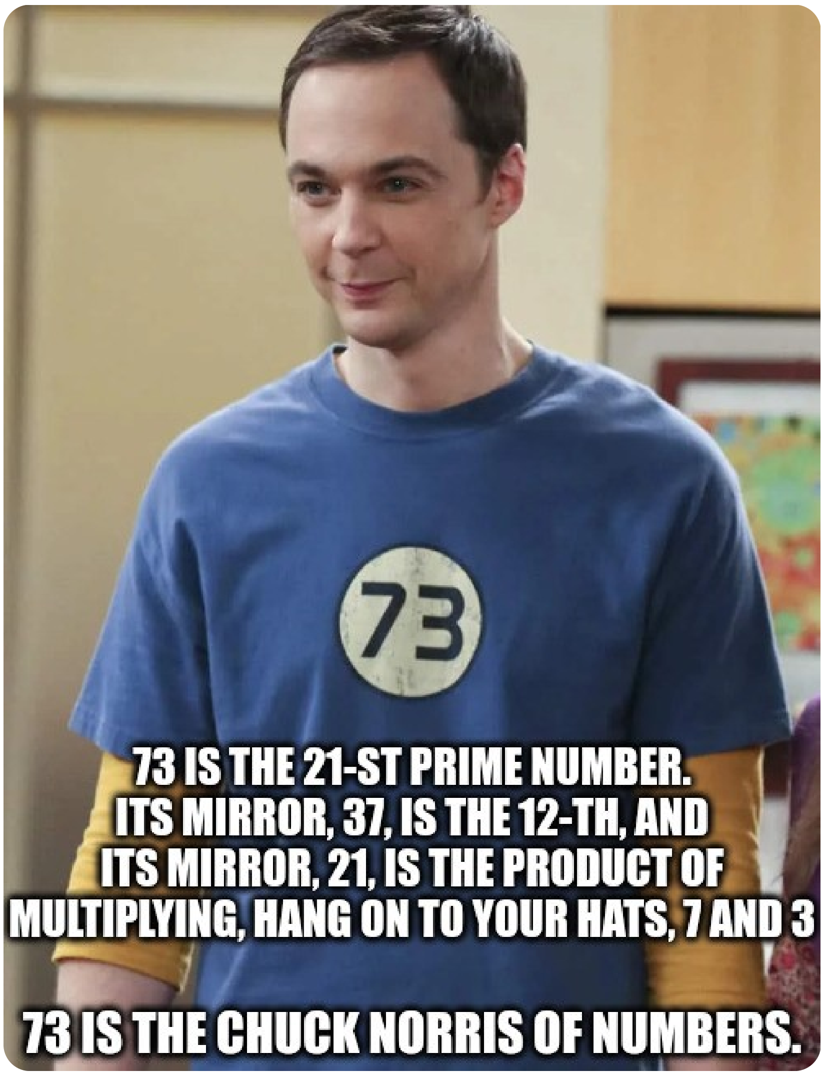
- ✓ the absolute value of the numbers: `2`, `-2` and `-3e4` ;
- ✓ the maximum value of `[-42, 3e2, 666]`.

Task 1.2

Apply the `min()` built-in function on the string `Beautiful is better than ugly.`.
What do you observe? What do you make of it?

Task 1.3

Using a built-in function, computes the value of 73 to the power 73.


73
73 IS THE 21-ST PRIME NUMBER.
ITS MIRROR, 37, IS THE 12-TH, AND
ITS MIRROR, 21, IS THE PRODUCT OF
MULTIPLYING, HANG ON TO YOUR HATS, 7 AND 3
73 IS THE CHUCK NORRIS OF NUMBERS.

Task 1.4

What are the built-in function that returns:

- ✓ `True` if there is at least one true element in a list?
- ✓ `True` if there is zero false element in a list?

Task 1.5

Using a built-in function, calculate the sum of `L1 + L2 + L3 + L4` where:

- ✓ `L1 = [1, 2, 3, 4]`;
- ✓ `L2 = [5, 7, 9, 32]`;
- ✓ `L3 = [23, 13, 17, 14, 16, 309]`;
- ✓ `L4 = [10, 20, 30, 40]`.

Task 1.6

Find an easy way to sort the names in this list `["Bob", "Emmett", "Gratton", "Mason"]`:

- ✓ from shortest to longest ;
- ✓ from longest to shortest.

Lambda function

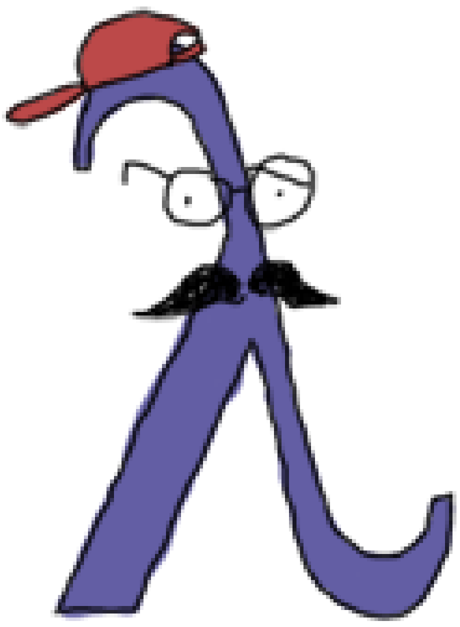
Task 2.1

Dig this piece of code. Try to figure its output.
Then, run it to see if you were right.

```
crazyFunction = lambda x, y: x * y
meaningOfLife = crazyFunction(6, 7)
print(meaningOfLife)
```

Task 2.2

Use the `sorted()` built-in function to build a new sorted list from list of lists, but by using the second value (count) inside each secondary list.
`animalsCounts = [['cat', 666], ['dog', 3], ['elephant', 42]]`.



Task 2.3

Dig this code `list(filter(lambda x: x > 10, [3.14, 101, 42, 666, -1]))`.
Try to predict its output. Then, run it to check if you're right.

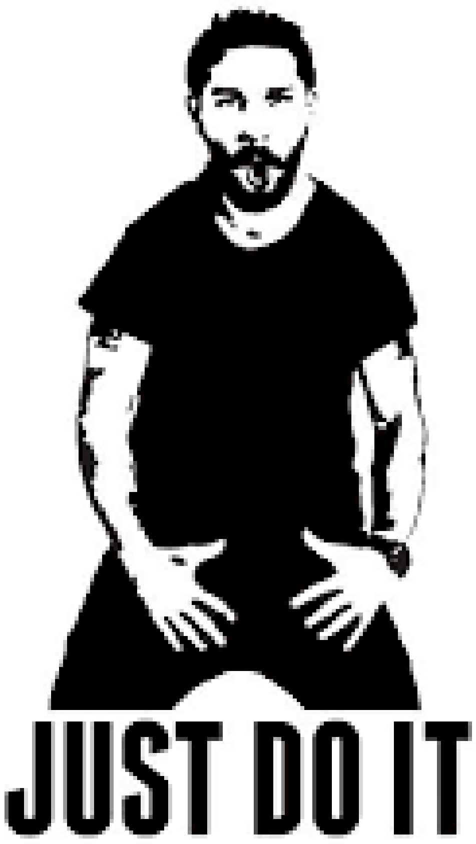
CHALLENGE

First, go back over the exercises from the previous subjects and categorize them:

- ✓ Which exercices felt easy? Which ones taught you the most?
- ✓ Which exercices were challenging? Which one was the biggest challenge?
- ✓ Which exercices would you like to revisit? Which one did you fail?
- ✓ Was there a moment when something ”clicked” for you?

Then, compare and share your previous classification with other students, in order to:

- ✓ Find someone to team with.
- ✓ Choose an exercise you both like to revisit Then, just do it again!
- ✓ Identify an exercise you both failed or didn't have time to try. Then, just do it!



- ✓ Finally, you must analyze, comment and improve each other's code.

This is called **code review**: it fosters higher code quality and knowledge sharing.

Magic built-in functions

Task 3.1

Test this code and try to explain it: `[*enumerate([42, 3, 4, 18, 3, 10])]`

Task 3.2

Create a function `check_even` that returns `True` if a number is even and `False` otherwise.
Use your previous function and the `filter()` built-in function in order to output a list of all even numbers contained in `[1, 2, 3, 4, 5, 6]`.

Task 3.3

Use `filter` to remove all strings with more than 4 characters from `['apple', 'banana', 'kiwi', 'pear']`.

Task 3.4

Apply `map` to convert this list of temperatures `[-10, 0, 17.6, 28, 100]`, from Celsius to Fahrenheit.

Task 3.5

Test this code and try to explain it:

```
first_names = ["Jackie", "Chuck", "Arnold", "Sylvester"]
last_names = ["Stallone", "Schwarzenegger", "Norris", "Chan"]
magic = [*zip(first_names, last_names[::-1])]

print(magic[0])
print(magic[1][0])
print(magic[1][1])
```

{EPITECH}

v 2.2

{EPITECH}