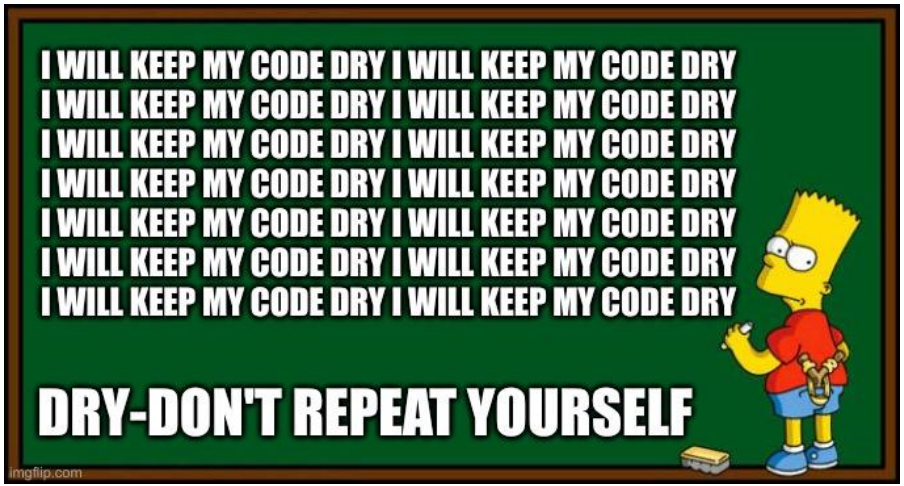
Functions in Python

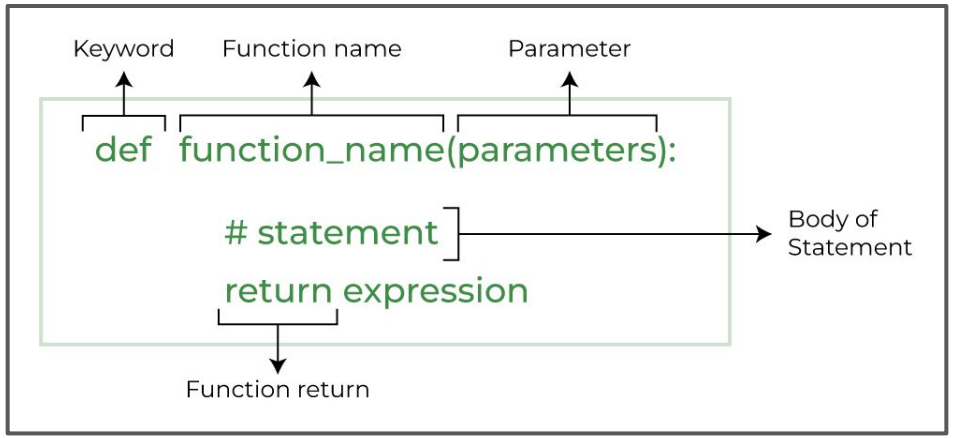
Dr. Sushil Sharma

What is a function?

- **A function is a block of code which <u>only runs</u> when it is called.**
- A function can return data as a result.
- A function helps avoiding code repetition.



How to define & call a function? - ***The* Syntax**




```
def my_function():
  print("Dzień Dobry from Krakow!")

my_function()
my_function()
my_function()
```

How to pass information into functions? - **The Arguments**

```
def my_function(fname):
  print(fname + " Gates")

my_function("Elon")
my_function("Bill")
my_function("Steve")
```

fname is a parameter.

“Elon” is an argument.

Parameters vs Arguments

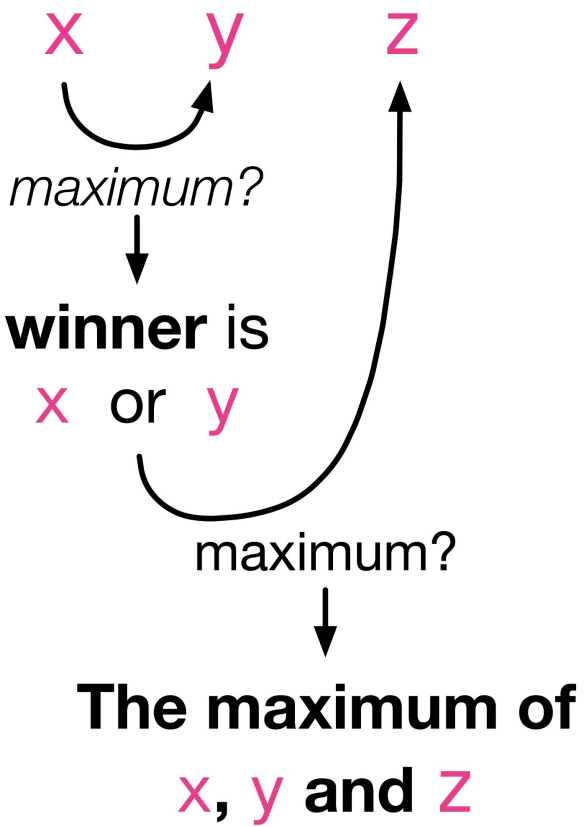
The terms *parameter* and *argument* can be used for the same thing: information that are passed into a function.

From a function's perspective:

A **parameter** is the variable listed inside the parentheses in the function definition.

An **argument** is the actual value that is sent to the function when it is called.

P1 - Write a Python function to find the maximum of three numbers.



How to provide default parameter values?

```
def my_function(name = "friend"):
  print("Hello", name)

my_function("Elon")
my_function("Bill")
my_function()
my_function("Steve")
```

```
def my_function(country = "Poland"):
  print("I am from", country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Passing & Returning Different Data Types

```
def my_function(person):
  print("Name:", person["name"])
  print("Age:", person["age"])

my_person = {"name": "Satyam", "age": 23}
my_function(my_person)
```

```
def my_function():
  return ["apple", "banana", "cherry"]

fruits = my_function()
print(fruits[0])
print(fruits[1])
print(fruits[2])
```

P2 - Write a Python function to sum all the numbers in a list.

Sample List : [8, 2, 3, 0, 7]

Expected Output : 20

P3 - Write a Python function to multiply all the numbers in a list.

Sample List : [8, 2, 3, -1, 7]

Expected Output : -336

P4 - Write a Python program to reverse a string.

Sample String : "1234abcd"

Expected Output : "dcba4321"

Keyword (*kwargs*) VS Positional Arguments

‚By name not position'

*ordering doesn't matter

```
def my_function(animal, name):
  print("I have a", animal)
  print("My", animal + "'s name is", name)

my_function(name = "Buddy", animal = "dog")
```

*ordering matters

```
def my_function(animal, name):
  print("I have a", animal)
  print("My", animal + "'s name is", name)

my_function("Buddy", "dog")
```

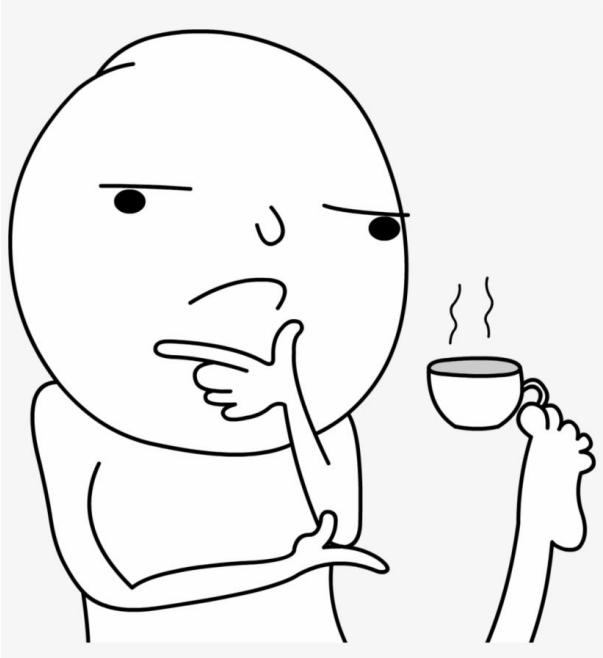
Mixing Positional & Keyword Arguments

```
def my_function(animal, name, age):
  print("I have a", age, "year old", animal, "named", name)

my_function("dog", name = "Buddy", age = 5)
```

*Positional arguments must come before keyword arguments.

P5 - **Can you** write a function that can take any number of numeric arguments and return their total?



*args and **kwargs - Allowing functions to accept unknown number of arguments

```
def my_function(*kids):
  print("The youngest child is " + kids[1])

my_function("X Æ A-12", "Techno", "Kai")
```

a *tuple* of arguments

Now, try to solve P5?

```
def my_function(*numbers):
  total = 0
  for num in numbers:
      total += num
  return total

print(my_function(1, 2, 3))
print(my_function(10, 20, 30, 40))
print(my_function(5))
```

```
def my_function(**myvar):
  print("Type:", type(myvar))
  print("Name:", myvar["name"])
  print("Age:", myvar["age"])
  print("All data:", myvar)

my_function(name = "Pawel", age = 95, city = "Zakopane")
```

```
*  for positional arguments
** for keyword arguments
```

Combining *args & **kwargs

The order must be:

1. regular parameters
2. *args
3. **kwargs

```
def my_function(title, *args, **kwargs):
  print("Title:", title)
  print("Positional arguments:", args)
  print("Keyword arguments:", kwargs)

my_function("User Info", "Sushil", "Sharma", age = 30, city = "Krakow")
```

More Problems

P6 - Write a Python function that checks whether a passed string is a palindrome or not.

Note: A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam .

P7 - Write a Python function that takes a number as a parameter and checks whether the number is prime or not.

Note : A prime number (or a prime) is a natural number greater than 1 and that has no positive divisors other than 1 and itself.

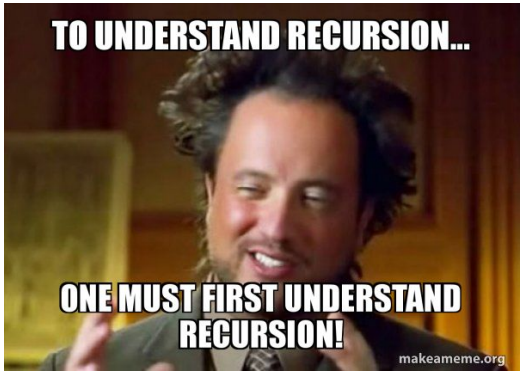
Recursion - A function calling itself

When written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

Every recursive function must have two parts:

- **A base case** - A condition that stops the recursion
- **A recursive case** - The function calling itself with a modified argument

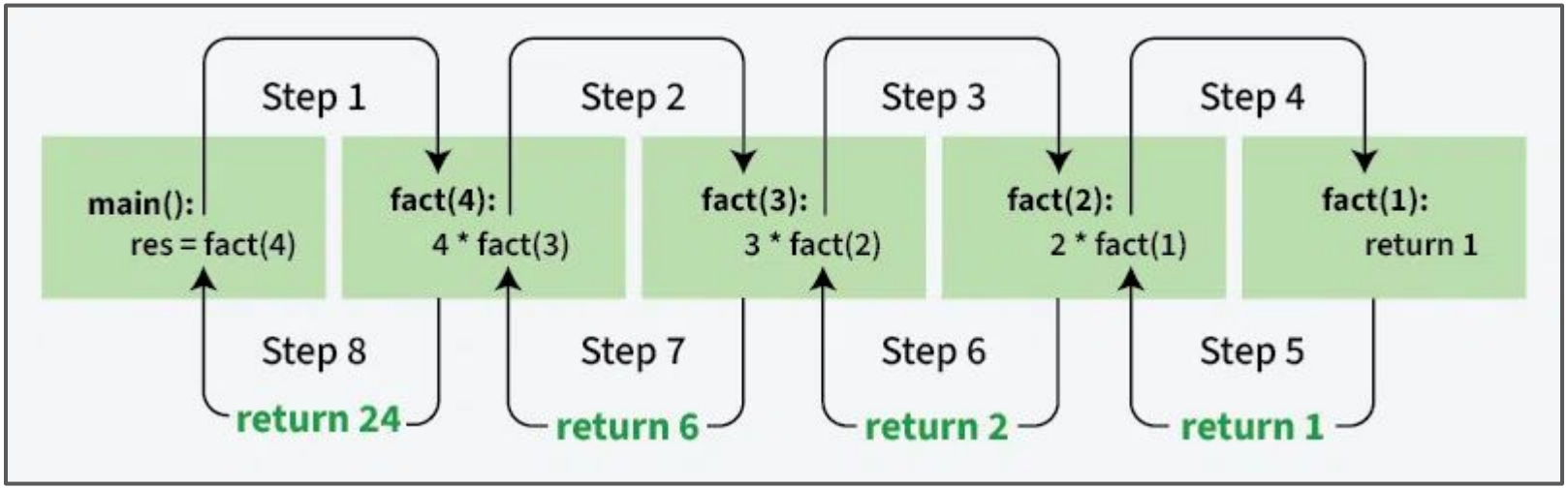
Without a base case, the function would call itself forever, causing a stack overflow error.



Example: Factorial of a number

```python
def fact(n):
  # Base case
  if n == 0 or n == 1:
    return 1
  # Recursive case
  else:
    return n * fact(n - 1)

print(fact(4))
```



P8 - Sum of Natural Numbers (n=3)

Input : n = 3

Output : 6

Explanation : The sum of first 3 natural numbers is 1+2+3 = 6.

Base Case: At n == 1, it returns 1 for n = 3, the recursion reaches this after going through 3 → 2 → 1.

Recursive Case: Each call adds n to sum(n-1) , so sum(3) = 3 + sum(2), sum(2) = 2 + sum(1).

P9 - Fibonacci with Recursion

Write a program and recurrence relation to find the Fibonacci series of n where n >= 0.

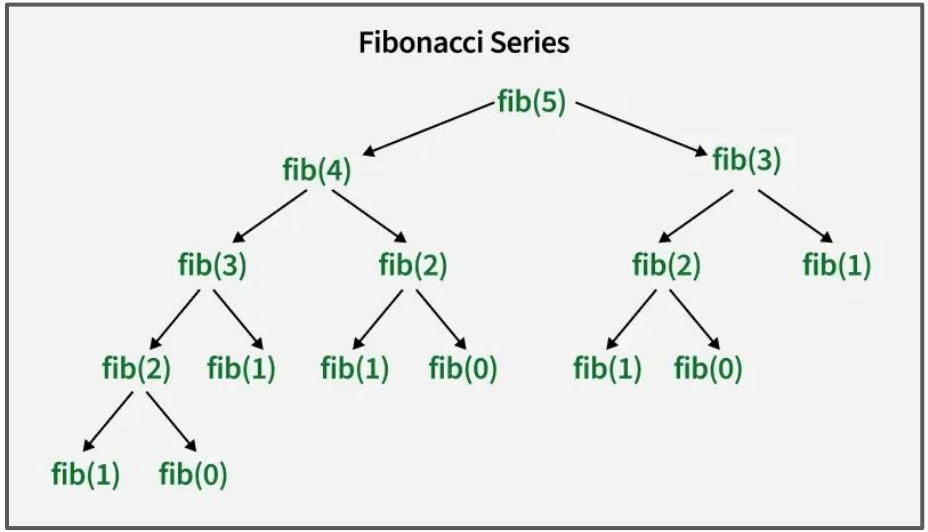
Mathematical Equation:

```
n if n == 0, n == 1;

fib(n) = fib(n-1) + fib(n-2) otherwise;
```

Recurrence Relation:

```
T(n) = T(n-1) + T(n-2) + O(1)
```



For more interesting problems: https://www.geeksforgeeks.org/dsa/recursion-practice-problems-solutions/

Lambda Function: A small anonymous function

A lambda function can take any number of arguments, but can only have one expression.

`lambda arguments : expression`

```
x = lambda a : 2025 - a
print(x(2002))
```

```
x = lambda a, b : a % b
print(x(51, 2))
```

```
x = lambda a, b, c : a**2 + b**2 - c**2
print(x(3, 4, 5))
```

Example - Sorting in Python

You have a list of dictionaries representing students:

```
students = [
    {"name": "Aisha", "score": 92},
    {"name": "Leo", "score": 85},
    {"name": "Mina", "score": 92},
    {"name": "Zed", "score": 70}
]
```

Without writing a full function definition, how can you sort this list so that:

1. Students with **higher scores come first**, and
2. If two students have the same score, the one whose **name comes alphabetically first** appears earlier?

```
sorted(students, key=lambda s: (-s["score"], s["name"]))
```

Scope of a function in Python

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

```
x = 300

def myfunc():
  x = 200
  print(x)

print(x)

myfunc()

print(x)
```

```
def myfunc():
  global x
  x = 300

myfunc()

print(x)
```

If you need to create a global variable, but are stuck in the local scope, you can use the `global` keyword.

How to take user input in python? - **Using `input()` function**

```
name = input("Enter your name:")
print(f"Hello {name}")
```

Python stops executing when it comes to the `input()` function, and continues when the user has given some input.

```
name = input("Enter your neighbour’s name:")
print(f"Hello {name}")
fav1 = input("What is his/her favorite animal:")
fav2 = input("What is his/her favorite color:")
fav3 = input("What is his/her favorite number:")
print(f"Gift him/her a {fav2} {fav1} with {fav3} legs?")
```

Announcement - *Home Assignment*

- The assignment counts for 20% of the total course grade.
- You **can** form a group of two students.
- You will create a presentation explaining your chosen project (covering what it is about, how it works, and any relevant details).
- You may choose any project topic you find interesting; a list of suggestions will be uploaded on Pegaz/Email.
- We will send Google Sheet link, you must fill your chosen topic and the name of your group partner by **8th December 2025**.
- The final submission deadline for the presentation is **12th December 2025** and you all will get to present it on **19th December** (online).

Thank You - Do Widzenia! ^_^