 # A function in Python is a block of reusable code that performs a specific task.

 # You can define a function once and use (call) it whenever needed — without writing the same code again and again.

```
def greet():
    print("Hello, welcome to Python!")

greet()
```

 # Functions with Parameters (Inputs):

```
def greet(name):
    print("hello", name)

greet("Atharva")
```

 # When you call a function, sometimes you want to give it some information (like numbers, names, etc.) to work with. That information is passed using parameters or arguments.

 # Parameter - The variable name inside the function definition

 # Argument - The actual value you pass to the function when calling it

 # Functions with Return Values:

```
def add(num1 , num2):
    print(num1 + num2)

add(10,20)
```

 # Arbitrary Arguments

 # Sometimes, you don’t know how many arguments someone might pass to your function.To handle this, Python gives us *args.

```
def greet(*name):
   print("hello", name)

greet("Atharva","akshay","santosh","riya")
```

 # Keyword Argument:

 # A keyword argument is an argument passed to a function using the key = value format

```
def student(name , age , grade):
   print("details of the student: ", name , age , grade)

student(name = "Atharva", age = 20, grade = "A" )
```

 # What is **kw args in Python?

 # If you don’t know how many keyword arguments (i.e., key=value pairs) will be passed to your function, you can use **kwargs.

 # it will give me it in dictionary form

```
def student(**kwargs):
   print(kwargs)
student(name = "Atharva", age = 20, grade = "A", contact = 9820919318)
```

 # A default parameter value is a value that is automatically used by a function if no argument is provided for that parameter during the function call.

```
def student(name = "santosh"):
   print(name)


student("Atharva")
student("Akashay")
student()
```

 # assing a List as an Argument (Simple Explanation):

 # You can give a list to a function just like any other value

```
def fruits(fruits_name):
   print(fruits_name)


fruits(["apple", "orange", "watermelon", "lemon"])


def my_function(fruits):
   for fruit in fruits:
      print(fruit)


my_function(["apple", "banana", "mango"])
```

 # Use the return statement inside the function to send the result back to the place where the function was called.

```
def add(num1 , num2):
   sum = num1 + num2
```

```
    return sum

result = add(5,10)
print(result)
```

 # In Python, you can’t leave a function, loop, or class empty — it causes an error.Use the pass statement as a placeholder to avoid the error.

```
def student():
    pass

student()
```

 # poistional argument

```
def student(name , age):
    print(name , age)

student("atharva",20)
```

 # keyword argument

```
def student(name ,age):
    print(name , age)

student(name = "atharva", age = 20)
```

 # But when adding the , / you will get an error if you try to send a keyword argument:

```
def student(name , age , /):
   print(name , age)

student(name = "atharva",age = 20)
```

 # But when adding the * , you will get an error if you try to send a poistional argument:

```
def student(* , name , age, ):
   print(name , age)

student("atharva",20)
```

 # Combine Positional-Only and Keyword-Only

```
def my_function(a, b, /, *, c, d):
  print(a + b + c + d)

my_function(5, 6, c = 7, d = 8)
```

 # Recursion means:

 # A function that calls itself.

 # It keeps calling itself until it reaches a stopping condition (called the base case).

```
def show(n):
   if (n    == 0):
      return
```

```
    print(n)
    show(n-1)

show(5)

def show(n):
    if (n == 0):
        return
    print(n)
    show(n-1)

show(5)
```

5 , 4, 3, 2 , 1

 # A lambda function is a small, anonymous (unnamed) function in Python.

 # It is used when you need a simple function for a short time.

```
x = lambda a : a + 10
print(x(5))

x = lambda a, b : a * b
print(x(5, 6))

x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```