

FUNCTIONS

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

1.Function Declaration:-

It is defined using the def keyword.

```
Syntax:
def function_name(parameters):
    # function body
    # code to perform a task
return result # optional, used to return a value
```

- function_name : The name of the function.
- parameters: The input parameters that we are passed to the function.
- function body: The statements written within a function.

Example:

```
#To create a function
def my_function():
    print("This is a function")
```

2. Calling a function:

 To call the function, use the function name, followed by parentheses.

Syntax:

result = my_function(arguments)

Here, arguments are the values passed to the function, and result is the value returned by the function (if any).

Example: **#To call a function** def my_function(): print("This is a function") my_function() **Output:**

This is a function

3. Parameters:

Parameters are variables used in the function definition.

4.Arguments:

 Arguments are the actual values passed to the function when it is called.

```
Syntax :
    def functionName(arg1, arg2):
        # What to do with function
    functionName(valueForArg1, valueForArg2)
Example :
    def addNum(num1, num2):
        print(num1 + num2)
        addNum(2, 4)
Output:
```

5.Return statement:

- A function may have a return statement to send a result back to the caller. If there is no return statement, the function returns None by default.
- In Python, you can use the return keyword to exit a function so it goes back to where it was called. That is, send something out of the function.
- The return statement can contain an expression to execute once the function is called.

```
Syntax :
  return
Example :
  def multiplyNum(num1):
    return num1 * 8
  result = multiplyNum(8)
  print(result)
Output :
```

What's the code above doing?

- I defined a function named multiplyNum and passed it num1 as an argument
- Inside the function, I used the return keyword to specify that I want num1 to be multiplied by 8
- After that, I called the function, passed 8 into it as the value for the num1 argument, and assigned the function call to a variable I named result
- With the result variable, I was able to print what I intended to do with the function to the terminal

Advantages:

- Enables reusability and reduces redundancy.
- Makes a code modular.
- Provides abstraction functionality.
- The program becomes easy to understand and manage.
- Breaks an extensive program into smaller and simpler pieces

Disadvantages:

Programmers have less control over how they work and less

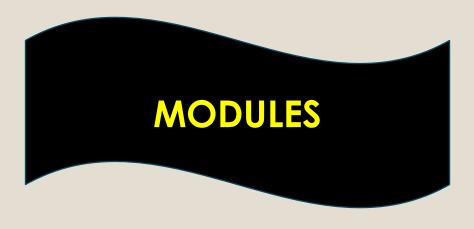
flexibility to customize their behaviour.

 Complexity: Using too many functions can make the code harder

to understand.

• Maintenance: Maintaining a large number of functions can be

challenging.



Modules:

- A module is a file containing python definitions and statements.
- It allows you to logically organize your python code.
- Modules help in code reusability and can be imported into other python scripts or modules.
- A module is simply a python file with a .py extension that can be imported inside another Python program.
- The name of the Python file becomes the module name.
- A module can contains
 - -Functions
 - -Classes
 - -Variables

1.Create a module:

 To create a module just save the code you want in a file with the file extension .py

<u>Syntax</u>:

mymodule.py

Example:

Save this code in a file named mymodule.py

```
def greeting(name):
```

```
print("Hello, " + name)
```

2.Import a module:

You can use the import statement to include a module in your script.

Syntax:

import module_name

Example:

import mymodule

mymodule.greeting("Rosy")

Output:

Hello, Rosy

3. Variables in Module:

 The module can contain functions, as already described, but also variables of all types (arrays, dictionaries, objects etc).

```
Syntax:
module_name.function_name()
Example:
Save this code in the file mymodule.py:
person1 = {
 "name": "John",
 "age": 36,
 "country": "Norway"
Import the module named mymodule, and access the person1 dictionary:
import mymodule
a = mymodule.person1["age"]
print(a)
```

Output:

4. Naming a module:

 You can name the module file whatever you like, but it must have the file extension .py

5.Renaming a module:

 You can create an alias when you import a module, by using the as keyword.

Syntax:

import module_name as function_name

Example:

Create an alias for mymodule called mx:

```
import mymodule as mx
a = mx.person1["age"]
```

print(a)

Output:

5.Built-in Modules:

• There are several built-in modules in Python, which you can import whenever you like.

Example:

• Import and use the platform module:

import platform

x = platform.system()

print(x)

Output:

Windows

6.Using the dir() function:

There is a built-infunction to list all the function names (or variable names) in a module.

Example:

• List all the defined names belonging to the platform module:

import platform

x = dir(platform)

print(x)

7.Import From Module:

You can choose to import only parts from a module, by using the from keyword.

Syntax:

```
from module_name import function_name
```

Example:

• The module named mymodule has one function and one dictionary:

```
def greeting(name):
 print("Hello," + name)
person1 = {
 "name": "John",
 "age": 36,
 "country": "Norway"
• Import only the person1 dictionary from the module:
from mymodule import person1
print (person1["age"])
Output:
```

Advantages:

- Reusability: Working with modules makes the code reusable.
- Simplicity: Module focuses on a small proportion of the problem, rather than focusing on the entire problem.
- Scoping: A separate namespace is defined by a module that helps to avoid collisions between identifiers.

Disadvantages:

Name Collisions: The variables, functions or classes should not

be with the same name.

Global state: The number of modules increases significantly,

which making it harder to manage and navigate through the

project.

• Complexity: Modules introduce global state which can be

problematic in larger codebases.