CIS 412 L Artificial Intelligence Lab
Topic - 04
Introduction to Python for Machine Learning

+88 01831-661534

jehadfeni@gmail.com

Python Functions

A function can be defined as the organized block of reusable code, which can be called whenever required.

- Python provide us various inbuilt functions like **range()** or **print()**. Although, the user can create its functions, which can be called user-defined functions.
- There are mainly two types of functions.
- **User-define functions** - The user-defined functions are those define by the **user** to perform the specific task.
- **Built-in functions** - The built-in functions are those functions that are **pre-defined** in Python.

Function syntax

- **Creating a Function**

Python provides the **def** keyword to define the function.

```
def my_function(parameters):
    function_block
return expression
```

- **Function Calling**

In Python, after the function is created, we can call it from another function. A function must be defined before the function call; otherwise, the Python interpreter gives an error. To call the function, use the function name followed by the parentheses.

```
#function definition
def hello_world():
    print("hello world")
# function calling
hello_world()
```

Function syntax

- **The return statement:**
  The return statement is used at the end of the function and returns the result of the function. It terminates the function execution and transfers the result where the function is called. The return statement cannot be used outside of the function.

Arguments

- The arguments are types of information which can be passed into the function. The arguments are specified in the parentheses. We can pass any number of arguments, but they must be separate them with a comma.

Arbitrary Arguments, *args:

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Arguments cont...

- **Keyword Arguments:**

We can also send arguments with the *key = value* syntax.

```
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Arguments cont…

- **Arbitrary Keyword Arguments, **kwargs**

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: ** before the parameter name in the function definition.

This way the function will receive a dictionary of arguments, and can access the items accordingly

```
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

The Pass statement

- function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

```
def myfunction():
  pass
```

Recursion / recursive function

- Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.
-
```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result)
  else:
    result = 0
  return result

print("\n\nRecursion Example Results")
tri_recursion(6)
```