# FUNCTION AND CLASSES

# Python Classes/Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.
- Create a Class

**for example**: class a:

x = 5

# Python Inheritance

- Inheritance allows us to define a class that inherits all the methods and properties from another class.
- Parent class is the class being inherited from, also called base class.
- Child class is the class that inherits from another class, also called derived class.
- Create a Parent Class

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
    def printname(self):
        print(self.firstname, self.lastname)
x = Person("John", "Doe")
x.printname()
```

# Child class

- To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class
- Create a class named Student, which will inherit the properties and methods from the parent class
- **For example**: class Student(Person):

  pass
- Use the pass keyword when you do not want to add any other properties or methods to the class
- Python also has a super() function that will make the child class inherit all the methods and properties from its parent
- By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent

# Python Iterators

- An iterator is an object that contains a countable number of values.
- An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.
- Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods __iter__() and __next__().
- Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable containers which you can get an iterator from.
- All these objects have a iter() method which is used to get an iterator.

# OBJECTS

- Now we can use the class named a to create objects
- **for example**:p1 = a()

  print(p1.x)
- The __init__() function is called automatically every time the class is being used to create a new object.
- The __str__() function controls what should be returned when the class object is represented as a string.
- Objects can also contain methods. Methods in objects are functions that belong to the object.

# Let us create a method in the Person class

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
  def myfunc(self):
    print("Hello my name is " + self.name)
p1 = Person("John", 36)
p1.myfunc()
```

# 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.
- **Creating a Function**
- In Python a function is defined using the def keyword
- **Example**

```
def my_fun():
    print("Hello from a function")
```

# Function

- **Calling a Function**
- To call a function, use the function name followed by parenthesis
- **Example:**

```
def my_fun():
    print("Hello from a function")
my_fun()
```

## Arguments:

- Information can be passed into functions as arguments.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

# TYPES OF FUNCTIONS

Different Types of Functions in Python

- Built-in Functions
- User-defined Functions
- Recursive Functions
- Lambda Functions
- Higher-order Functions

# Built-in Functions

- These functions are built into the Python language and can be used without the need for additional code. Some examples of built-in functions are print(), len(), sum(), min(), max(), etc.
- **print() ->** Outputs a message to the console or standard output device.
- **input() ->** Takes user input from the console or standard input device.
- **len() ->** Returns the length of an object, such as a string, list, or tuple.
- **type() ->** Returns the data type of an object.
- **range() ->** Creates a sequence of numbers between the specified start and end points.
- **int() ->** Converts an object to an integer data type.
- **float() ->** Converts an object to a float data type.

- **str()** -> Converts an object to a string data type.
- **bool() ->** Converts an object to a boolean data type.
- **max() ->** Returns the maximum value in a list or sequence.
- **min() ->** Returns the minimum value in a list or sequence.
- **sum() ->** Calculates the sum of a list or sequence.
- **sorted() ->** Sorts a list or sequence in ascending order.
- **abs() ->** Returns the absolute value of a number.

# User-defined Functions

- User-defined functions are functions that the programmer creates to perform a specific task or set of tasks. Defining a function allows you to reuse code and makes your code more modular and easier to read
- In Python, you can define a function using the def keyword, followed by the function name and any parameters it requires in parentheses. The code block that makes up the function is indented beneath the def statement.
- def greet(name):
- print(f"Hello, {name}!")
- In this example, the greet() function takes one parameter, name, to personalize the greeting. When the function is called, it will print out a message to the console that greets the specified name.

# Recursive Functions

- Recursive functions in Python call themselves to perform a task repeatedly until a certain condition is met.
- Recursive functions can be used to solve problems that can be broken down into smaller sub-problems that can be solved using the same approach.
- **For example:**

```
def factorial(n):
        if n == 1:
          return 1
        else:
          return n * factorial(n-1)
print(factorial(5))
```

# Lambda Functions

- A lambda function is a small anonymous function.
- A lambda function can take any number of arguments, but can only have one expression.
- **Syntax:**

  lambda arguments : expression
- **For example:**

```
x = lambda a : a + 10
print(x(5))
```
- The power of lambda is better shown when you use them as an anonymous function inside another function.