Data Science & Analytics using Python on AWS Platform Day 7

[Chennai], [02.07.2024]

aspire SYSTEMS | attention. always.

AGENDA

DATA SCIENCE & ANALYTICS USING PYTHON ON AWS PLATFORM

BASICS OF PYTHON SYNTAX

Python is known for its simple and readable syntax. Here's a basic overview:

1. Printing Output
2. Comments
3. Variables and Assignment
4. Indentation
5. Basic Data Types
6. Collections
7. Control Structures
8. Functions

INTRODUCTION TO PYTHON

- Python is an interpreted, high-level, general-purpose programming language
- Created by Guido van Rossum and first developed in late 1980s
- Code readability and object-oriented approach
- Python is often described as a "batteries included" language due to its comprehensive standard library.

Interesting fact:

Python is named after the comedy television show **Monty Python’s Flying Circus**. It is not named after the Python snake.

How to Install python?

- Python 3.12 Version(Go for Latest Version)
- Visual Studio Code latest Version
- PyCharm community latest Version
- Jupitor Notebook

www.python.org
Go to Website and download Latest Version

Python Programming Basics

Python basic programming concepts like

1. Variables in Python Programming
2. Python Datatypes
3. Python Operators
    i) Arithmetic Operators
    ii) Assignment Operators
    ii) Relational/Comparision Operators
    iii) Logical operators
4. Conditional statements
5. Looping statements
6. Type conversion

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- Variables do not need to be declared with any particular type, and can even change type after they have been set.

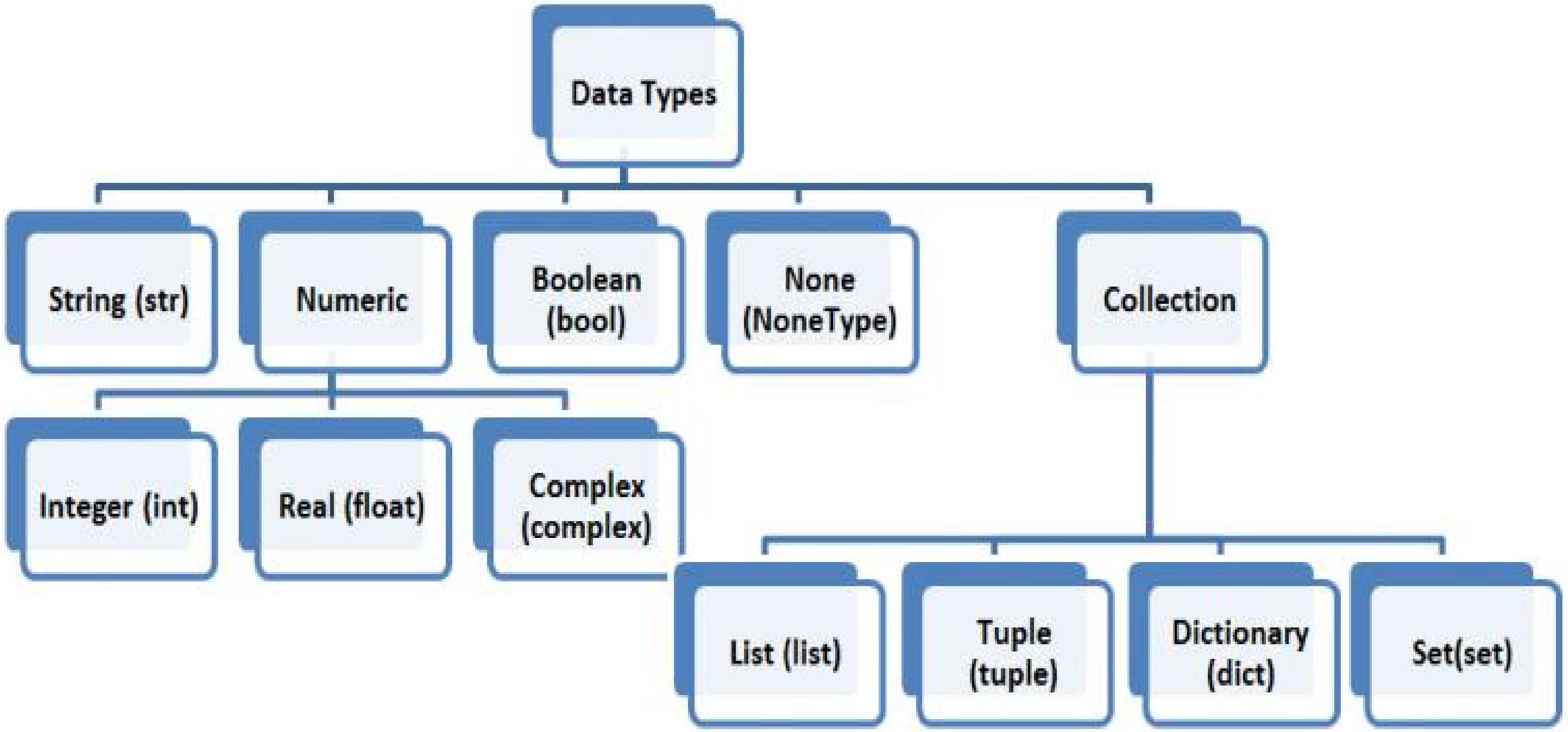
PYTHON DATA TYPES

Python Data types

Python has the following data types built-in by default, in these categories:

Text Type	:	str
Numeric Types	:	int, float, complex
Sequence Types	:	list, tuple, range
Mapping Type	:	dict
Set Types	:	set, frozenset
Boolean Type	:	bool
Binary Types	:	bytes, bytearray, memoryview
None Type	:	NoneType

Python Data types



EXAMPLES:

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 5+1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict

EXAMPLES:

Example	Data Type
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

COLLECTIONS

Python Lists

- Lists are one of the most powerful data structures in python.
- Lists are sequenced data types.
- In Python, an empty list is created using list() function.
- They are just like the arrays declared in other languages.
- But the most powerful thing is that list need not be always homogeneous.
- A single list can contain **strings, integers, as well as other objects.**
- Lists can also be used for implementing stacks and queues.
- **Lists are mutable**, i.e., they can be altered once declared.
- The elements of list can be accessed using indexing and slicing operations.

Python List Methods

Methods	Descriptions
append()	adds an element to the end of the list
extend()	adds all elements of a list to another list
insert()	inserts an item at the defined index
remove()	removes an item from the list
pop()	returns and removes an element at the given index
clear()	removes all items from the list
index()	returns the index of the first matched item
count()	returns the count of the number of items passed as an argument
sort()	sort items in a list in ascending order
reverse()	reverse the order of items in the list
copy()	returns a shallow copy of the list

Python Lists

- Python lists are one of the most versatile data types that allow us to work with multiple elements at once. For example,

Create Python Lists

1)List of programming languages

```
['Python', 'C++', 'JavaScript']
```

2) Empty list

```
my_list = []
```

3) List with mixed data types

```
my_list = [1, "Hello", 3.4]
```

4)ADD Item in list

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Python -Access Lists

Access List Elements

```
my_list = ['p', 'r', 'o', 'b', 'e']
```

1)First item

```
print(my_list[0])  # p
```

2) Third item

```
print(my_list[2])  # o
```

3) Fifth item

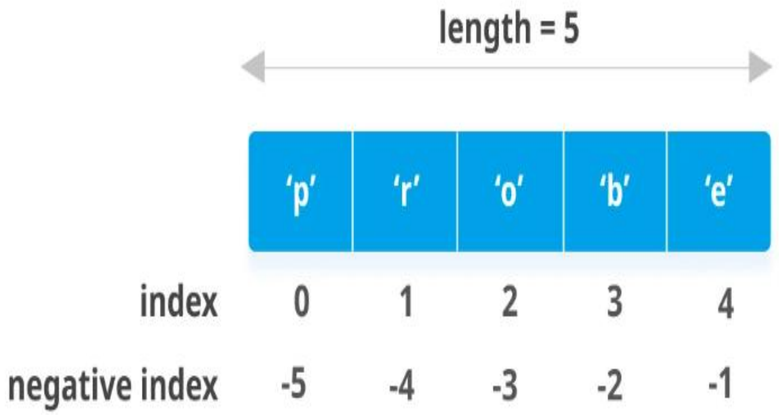
```
print(my_list[4])  # e
```

4) **Nested List**

```
n_list = ["Happy", [2, 0, 1, 5]]
```

5) **Nested indexing**

```
print(n_list[0][1])
print(n_list[1][3])
```



List Slicing - Python

We can access a range of items in a list by using the slicing operator

```
my_list = ['A','S','P','I','R','E','S','Y','S']
```

i) elements from index 2 to index 4

```
print(my_list[2:5])
```

ii)Elements from index 5 to end

```
print(my_list[5:])
```

ii)elements beginning to end

```
print(my_list[:])
```

Python Tuple

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

1)Creating a Tuple
2)Access Tuple Elements

There are various ways in which we can access the elements of a tuple.

- Indexing
- Negative Indexing
- Slicing

Python Tuple

3)Changing a Tuple

Unlike lists, tuples are immutable.
This means that elements of a tuple cannot be changed once they have been assigned. But, if the element is itself a mutable data type like a list, its nested items can be changed.

Tuple Methods:
Methods that **add items or remove items** are not available with tuple. Only the following two methods are available.
i) Count
ii) index

Python Dictionary

- Python dictionary is an unordered collection of items. Each item of a dictionary has a key/value pair.
- Creating a dictionary is as simple as placing items inside curly braces {} separated by commas.
- An item has a key and a corresponding value that is expressed as a pair (key: value).
- While the values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.

Python Dictionary Methods

Method	Description
clear()	Removes all items from the dictionary.
copy()	Returns a shallow copy of the dictionary.
fromkeys(seq[, v])	Returns a new dictionary with keys from seq and value equal to v (defaults to None).
get(key[,d])	Returns the value of the key. If the key does not exist, returns d (defaults to None).
items()	Return a new object of the dictionary's items in (key, value) format.
keys()	Returns a new object of the dictionary's keys.
pop(key[,d])	Removes the item with the key and returns its value or d if key is not found. If d is not provided and the key is not found, it raises KeyError.
popitem()	Removes and returns an arbitrary item (key, value). Raises KeyError if the dictionary is empty.
setdefault(key[,d])	Returns the corresponding value if the key is in the dictionary. If not, inserts the key with a value of d and returns d (defaults to None).
update([other])	Updates the dictionary with the key/value pairs from other, overwriting existing keys.
values()	Returns a new object of the dictionary's values
popitem()	Removes and returns an arbitrary item (key, value). Raises KeyError if the dictionary is empty.

Python Dictionary Methods

Removing elements from a dictionary

1)create a dictionary
squares = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

2) remove a particular item, returns its value
Output: 16
print(squares.pop(4))

3)Output: {1: 1, 2: 4, 3: 9, 5: 25}
print(squares)
remove an arbitrary item, return (key,value)

4) Output: (5, 25)
print(squares.popitem())

5)Output: {1: 1, 2: 4, 3: 9}
print(squares)

6)remove all items
squares.clear()

7) Output: {}
print(squares)

8)delete the dictionary itself
del squares

9)Throws Error
print(squares)

Python Sets

- A Set is an unordered collection of data types that is **iterable, mutable and has no duplicate elements**. The order of elements in a set is undefined though it may consist of various elements. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

Examples:

```
1)set1 = set()
print("Initial blank Set: ")
print(set1)
2)set1 = set("Aspire System")
print("\nSet with the use of String: ")
print(set1
3)set1 = set(["Aspire", "For", "Aspire"])
print("\nSet with the use of List: ")
print(set1)
```

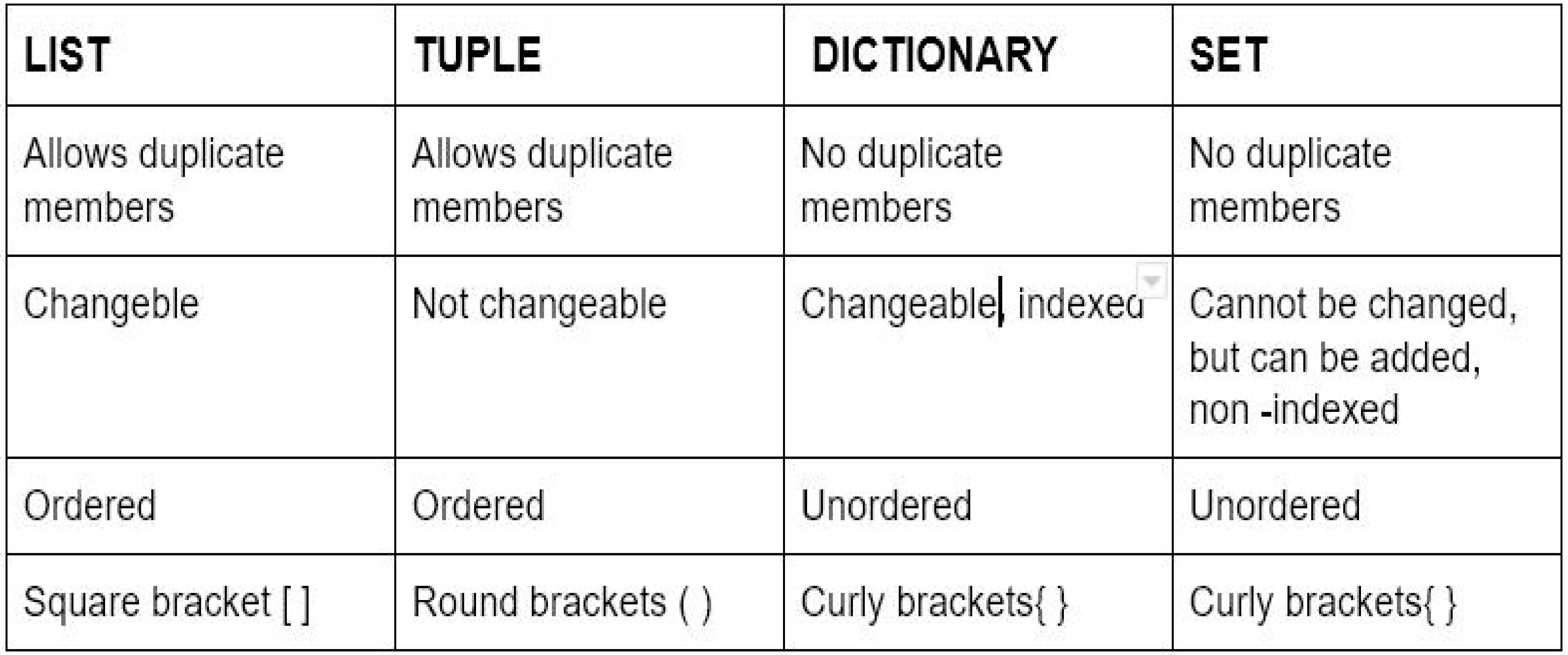
Python Collections Module

The collection Module in Python provides different types of containers. A Container is an object that is used to store different objects and provide a way to access the contained objects and iterate over them. Some of the built-in containers are **Tuple, List, Dictionary**, etc.
The different containers provided by the collections module are

Different containers

1. Counters
2. OrderedDict
3. DefaultDict
4. ChainMap
5. NamedTuple
6. DeQue
7. UserDict
8. UserList
9. UserString

Python Collections Module



LIST	TUPLE	DICTIONARY	SET
Allows duplicate members	Allows duplicate members	No duplicate members	No duplicate members
Changeble	Not changeable	Changeable, indexed	Cannot be changed, but can be added, non -indexed
Ordered	Ordered	Unordered	Unordered
Square bracket [ ]	Round brackets ( )	Curly brackets{ }	Curly brackets{ }

Python Keywords

Keywords in Python are reserved words that can not be used as a variable name, function name, or any other identifier.

List of all keywords in Python

The list of keywords is :
[‘False’, ‘None’, ‘True’, ‘and’, ‘as’, ‘assert’, ‘async’, ‘await’, ‘break’, ‘class’, ‘continue’, ‘def’, ‘del’, ‘elif’, ‘else’, ‘except’, ‘finally’, ‘for’, ‘from’, ‘global’, ‘if’, ‘import’, ‘in’, ‘is’, ‘lambda’, ‘nonlocal’, ‘not’, ‘or’, ‘pass’, ‘raise’, ‘return’, ‘try’, ‘while’, ‘with’, ‘yield’]

Printing all keywords at once using "kwlist()"

```
import keyword
print(keyword.kwlist)
```

User Input in Python

1. Read String from Console
2. Read Multiple Strings from User through Console

- Python allows for user input.
- That means we are able to ask the user for input.
- The method is a bit different in Python 3.6 than Python 2.7. Python 3.6 uses the **input() method**.
- Python 2.7 uses the **raw_input() method**.

String Manipulation

```
# String Manipulation
'''
 S  a  m  p  l  e
 0  1  2  3  4  5
-6 -5 -4 -3 -2 -1
'''
```

```
s = "sample"
print(s)
print(s[0:2])
print(s[:5])
print(s[1:])
print(s[-1])
print(s[-2:-1])
print(s[:-1])
print(s[::-1])
```

Conditional Statement

1. IF Statement
2. IF-ELSE Statement
3. IF ELIF Statement
4. NESTED IF Statement

1. for loop
2. for loop with else
3. Range function
4. While loop and
5. While loop with else

Conditional Statement

1) if Statement

```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")
```

2) if...else Statement

```
num = 3
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Conditional Statement

3) if...elif...else Statement

```
Val1 = 200
Val2 = 33
if Val2 > Val1:
  print("b is greater than a")
elif Val1 == Val2:
  print("Val1 and val2 are equal")
else:
  print("Val1 is greater than Val2")
```

4) Nested if Statement

```
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

Functions

- Function names have the **same rules as variable names**. The function is a block of related statements designed to perform a computational, logical, or evaluative task.
- The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to **reuse code** contained in it over and over again.

Syntax:

```
def function_name(parameters):
    statements()
```

Function -Components

Function consists of the following components.

- Keyword def that marks the start of the function header.
- A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
- Parameters (arguments) through which we pass values to a function. They are optional.
- A colon (:) to mark the end of the function header.
- Optional documentation string (docstring) to describe what the function does.
- One or more valid python statements that make up the function body. Statements must have the same indentation level (usually 4 spaces).
- An optional return statement to return a value from the function.

Calling a Functions

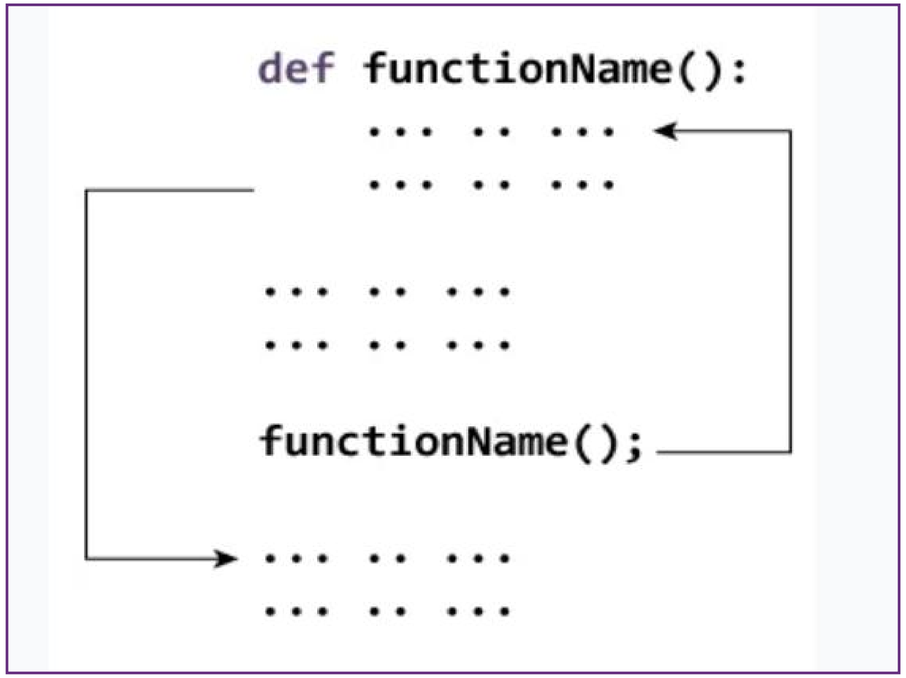
- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

Example:

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

my_function()
```

Working Function in Python



```
def functionName():
    ... .. ...
    ... .. ...

... .. ...
... .. ...

functionName();

... .. ...
... .. ...
```

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.

Parameters or Arguments:

From a function's perspective:

- A parameter is the variable listed inside the parentheses in the function definition.
- An argument is the value that is sent to the function when it is called.

Function Arguments

A value passed to a function (or method) when calling the function

- Default Arguments
- Keyword Arguments
- Arbitrary Arguments or variable length arguments
  - Arbitrary Arguments, *args
  - Arbitrary Keyword Arguments, **kwargs

(add two asterisk: **)

Function -Pass Statement

The pass statement is a **null statement**.The pass statement is generally used as a placeholder i.e. when the user does not know what code to write.

So user simply places pass at that line. Sometimes, pass is used when the user doesn’t want any code to execute. So user can simply place pass where empty code is not allowed, like in loops, function definitions, class definitions, or in if statements. So using pass statement user avoids this error.

Syntax:

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

File Handling

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files.

Working of open() function

Syntax:
f = open(filename, mode)

Where the following mode is supported:

r: open an existing file for a read operation.
w: open an existing file for a write operation. If the file already contains some data then it will be overridden.
a: open an existing file for append operation. It won’t override existing data.
 r+: To read and write data into the file. The previous data in the file will be overridden.
w+: To write and read data. It will override existing data.
a+: To append and read data from the file. It won’t override existing data.

File Handling(cont..)

1)Read Only Parts of the File

2)Readline()

3)Readlines()

4)By looping through the lines of the file, you can read the whole file, line by line

5)Close Files

6)Create New File using open Function

Working of open() function

Syntax:
f = open(filename, mode)

File Handling (Examples)

1)Read Method:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

2)Readline:

```
f = open("demofile.txt", "r")
print(f.readline())
```

3)Readlines(Read 2 lines)

```
f = open("demofile.txt", "r")
print(f.readlines())
```

4)Close files

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

5)Write to an Existing file

```
f = open("demofile2.txt", "a")
f.write("Now the file has more
content!")
f.close()
```

6)Create New file

```
f = open("myfile.txt", "x")
```

File Handling (Examples)

7) Delete a File:

To delete a file, you must import the OS module, and run its os.remove() function.

```
import os
os.remove("demofile.txt")
```

8) Check if File exist:

To avoid getting an error, you might want to check if the file exists before you try to delete it:

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

9) Delete Folder

To delete an entire folder, use the os.rmdir() method:

```
import os
os.rmdir("myfolder")
```

Modules or Packages

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables.

1)Import Module in Python – Import statement
2)The from import Statement

1)We can import the functions, classes defined in a module to another module using the import statement in some other Python source file.

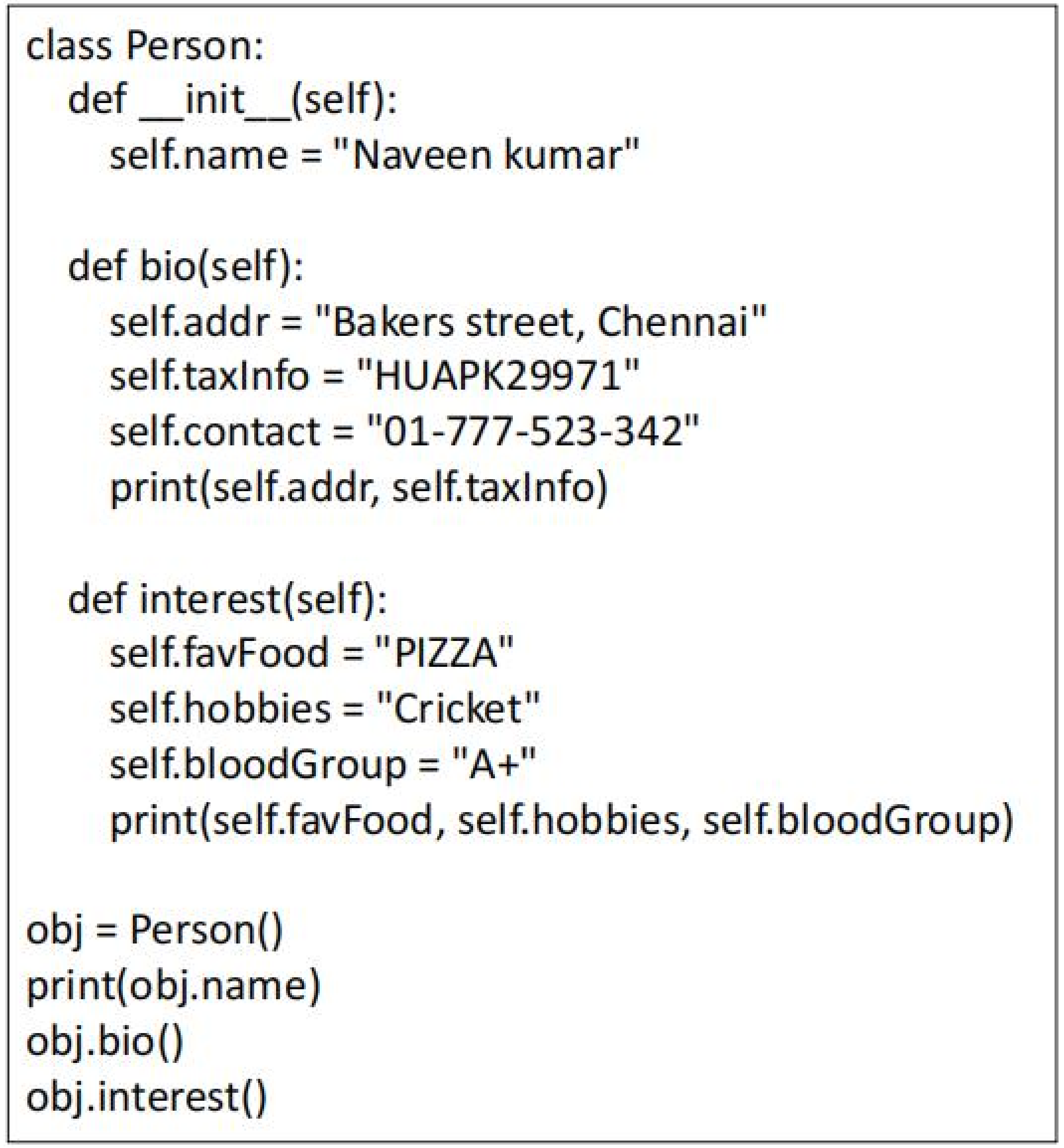
Syntax:

```
import module
```

Modules or Packages

Methods

```
1) import customer_list
    a=customer_list.Person
    a.interest()
2) from customer_list import Person
    a=Person
    a.interest()
3) from customer_list import interest as C1
   a=C1
   a.interest()
```



```
class Person:
    def __init__(self):
        self.name = "Naveen kumar"

    def bio(self):
        self.addr = "Bakers street, Chennai"
        self.taxInfo = "HUAPK29971"
        self.contact = "01-777-523-342"
        print(self.addr, self.taxInfo)

    def interest(self):
        self.favFood = "PIZZA"
        self.hobbies = "Cricket"
        self.bloodGroup = "A+"
        print(self.favFood, self.hobbies, self.bloodGroup)

obj = Person()
print(obj.name)
obj.bio()
obj.interest()
```

a t t e n t i o n.
a l w a y s.

Questions?

aspire SYSTEMS | attention. always.

Thank you

naveen.subramaniam@aspiresys.com

aspire SYSTEMS | attention. always.