INTERNSHIP ON DATA SCIENCE WITH PYTHON

SIDE A

- Basics of Python
- Lists
- Tuples
- Dictionaries
- Numpy
- Pandas
- Matplotlib
- Webscraping
- Case Study on Numpy
- Project on Pandas (Predicting Covid 19 cases in India)

UNIQUE IN THIS PROGRAM

Python for Data Science

- Python is a widely used programming language which is highly preferred for Data Science.
- It offers many libraries for analysis purpose that we are going to cover in our program:
  - Pandas
  - NumPy
  - Matplotlib
  - Seaborn
  - Scikit-learn

Introduction to Python

How are Tech Giants using Python?

How are Tech Giants using Python – Google

- Python is an official language at Google
- Python and C++ are used to write the core search algorithms at Google
- YouTube: Previously written in PHP (Hypertext Preprocessor) but eventually got replaced by Python
- Google uses Python to implement Machine Learning, AI as well as robotics projects
- Google uses SciPy, Scikit, NumPy, and various kinds of notebook UIs for performing data analysis

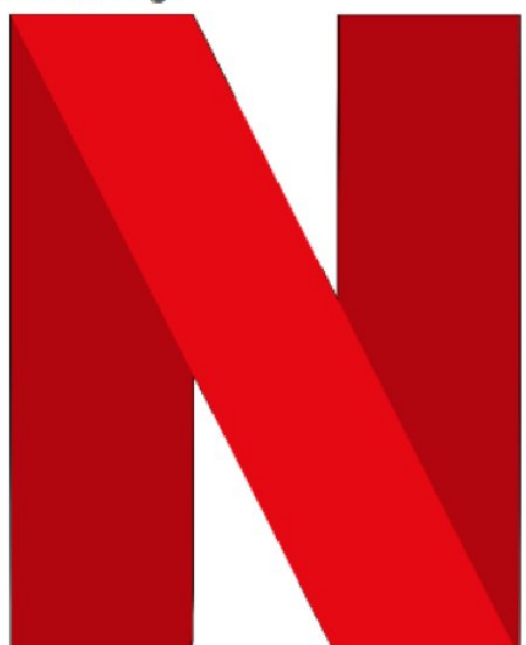
Google Logo

Until 2012, the designer and creator of Python, Guido van Rossum was employed by Google, when he left to work for Dropbox

How Tech Giants are using Python – Netflix

- Netflix uses Python for enhancing machine learning capabilities that analyze movies, optimize streaming, and pull-out images to display thumbnails
- Operations: Netflix uses Python libraries like NumPy and SciPy to perform numerical analysis
- Security: Python for security automation, risk classification, remediation, and vulnerability identification

Netflix Logo



Fundamentals of Python

Fundamentals of Python – Syntax Rules

01 Python is case sensitive. **Hello** is not equal to **hello**

02 Python does not have a **command terminator**

03 Each line can have one statement at most. For multiple statements, you should use **semicolon ;**

04 **Comments** in Python start with **'#'** or **''''**. For 1- line comments, we use **'#'** and for multiline use **''''** **(triple quotes either single or double)**

05 Python supports **multiline statements,** i.e., **Line Continuation**. To it, we put **backslash** '\' at the end of each line

Calculator

Python Token

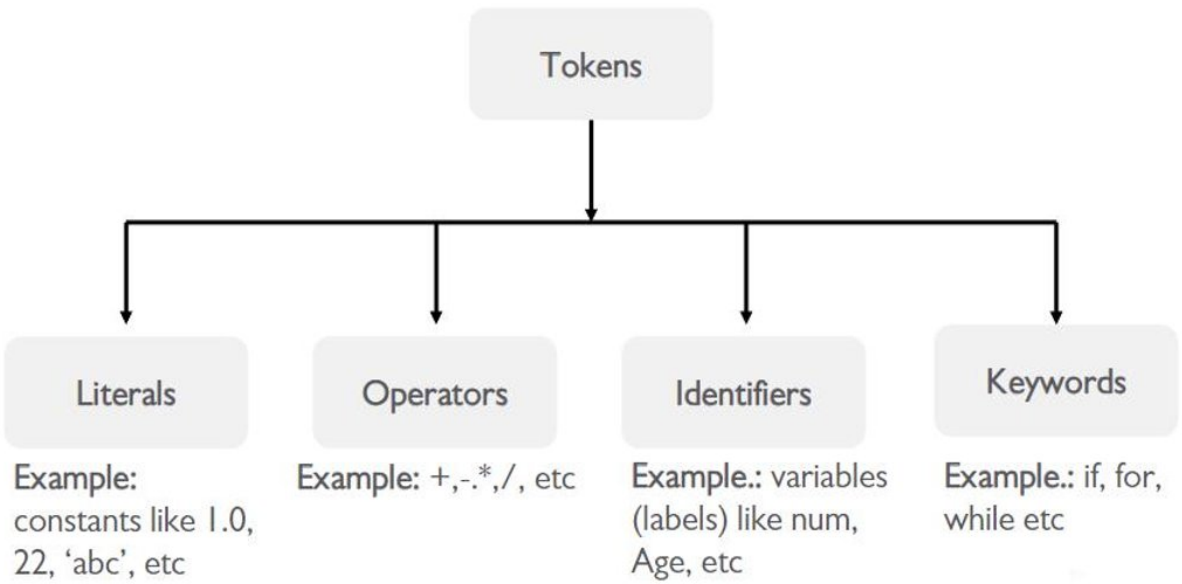
Tokens in Python

Python breaks every logical line into a set of elementary lexical elements known as **Token.**

Code Example

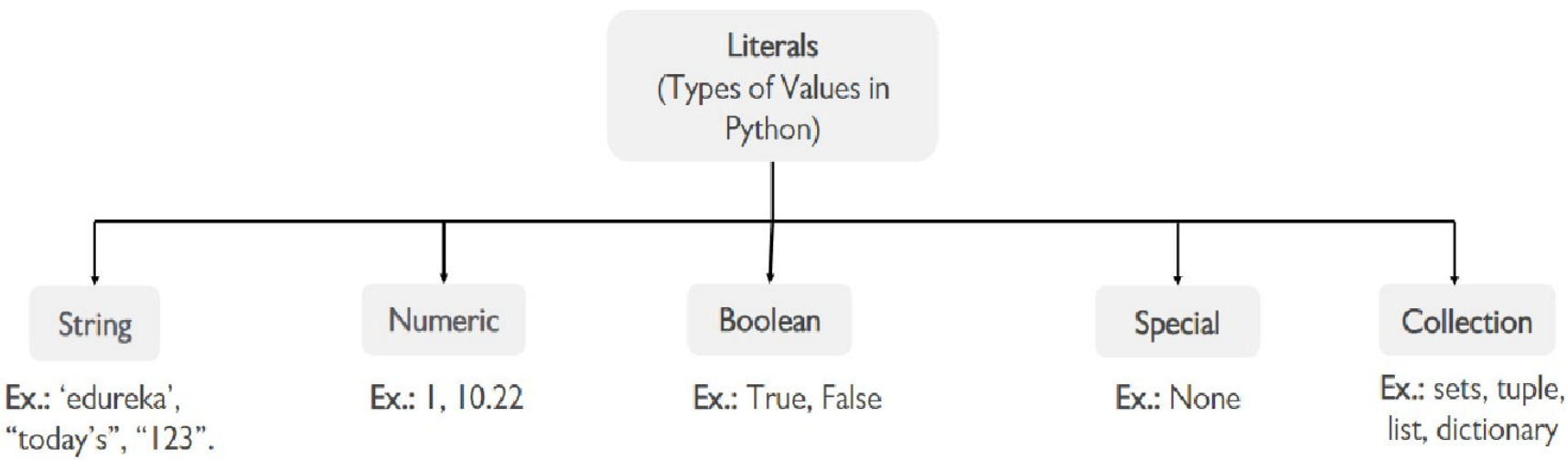
```
# Sum of two numbers in Python
a = 10
b = 20
print(a+b)

# Here 10,20 are literals, stored in
variable(identifier) a, b.
print is a keyword and + is an
operator
```

Tokens in Python: Types



Literals in Python

Literals are values used in Python



In other words, literals are data assigned to a variable or a constant

Keywords in Python

Keywords are set of reserved words with some specific functionality

Terminal command for keywords

```
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

Identifier in Python – Just like Labels

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another

Assigning values *10* and *edureka!* to variables *var* and *Var* respectively

Code Example

```
var = 10
Var = 'edureka!'
print(var,Var)
```

OUTPUT

```
10 edureka!
```

Operators and Variables in Python

Use Case1 – Build a Basic Calculator

Use Case1 – Building a Basic Calculator

Problem Statement: Write a Python code to build a basic calculator

NOTE: Input must be taken from the user to perform various arithmetic operations

Operators in Python

Operators are used to perform operations on the operands

For example, 2 + 3 = 5, here 2 and 3 are operands and + is called operator

Check out the demo from the LMS

- Arithmetic: +, -, /, *, %, //, **
- Comparison: <, >, !=
- Bitwise: &, |, >>, <<, ~, ^
- Membership: in, not in
- Assignment: =, +=, -=, *=, /=
- Logical: and, or
- Identity: is, is not

Variables

Variables are reserved memory locations to store values

Assigning values 10 and edureka! to variables A and B respectively

Code example

```
1  A=10
2  B='EDUREKA'
3  print(A,B)
```

Code output

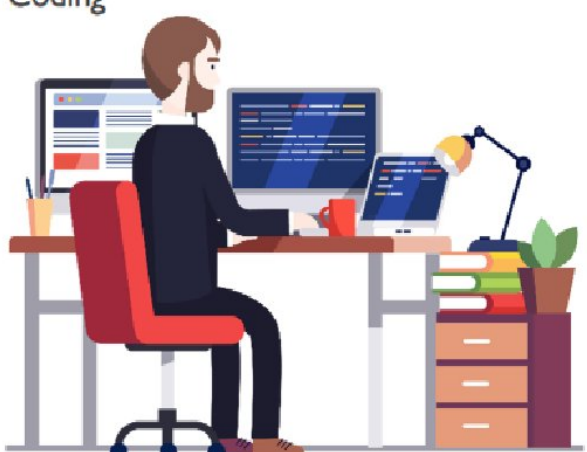
```
10 EDUREKA
```

- Variable cannot use any special character except underscore
- Keyword cannot be a variable name
- Variable cannot start with a number

Use Case 2 – Multiplication Table

Problem Statement: Write a python code to print the multiplication table of user's choice

Coding



Expected code output

```
1   n = int(input("Enter a number "))
2   for i in range(1, 11):
3     print(n,"X",i,"=",n*i)
```

```
Enter a number 7
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
7 X 10 = 70
```

NOTE:

- `input()` is taken from the user of type integer
- `for` is looping statement, will be covered in subsequent modules

Input from the User – Illustration 2

- Imagine you are writing a program for cricket game where you give liberty to the user to choose his players
- User enters all three values together, using `split()` function value is assigned to individual variable

Code example

```
All_rounders, Batsmen, Bowlers = input("Enter three values: ").split()
print("Number of All-rounders : ", All_rounders)
print("Number of Batsmen are : ", Batsmen)
print("Number of Bowlers are : ", Bowlers)
print()
```

Code output

```
Enter a three value: 3 4 4
Number of All-rounders :  3
Number of Batsmen are :  4
Number of Bowlers are :  4
```

Output – Notice multiple input entered

Functions in Python I

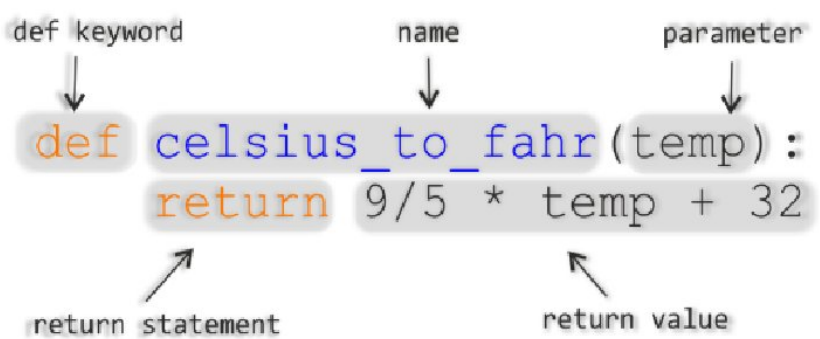
Functions in Python

- Functions in Python are a group of related statements that performs a specific task and can be called repeatedly
- They make extensive programs more organized and manageable

Syntax:

```
def
function_name(parameters):
   """docstring"""
   statement(s)
```

Add docstring to define what a function does



NOTE: Click here to learn about **built-in python function**

Function Definition and Function Call

Function Definition

The set of statements are written once and executed multiple times

Function Call

Once we define the function, we can call it multiple times

Code example

```
def print_name(str):
    print("Welcome to Python, ",str)
    return()

str = input("Enter your name :")
print_name(str)
```

Code output:

```
Enter your name : Jack
Welcome to Python, Jack
```

The Return Statement

The Return statement terminates the execution of a function and returns control to the calling Function

Code example

Return Statement

Function Call

```
def add(a,b):
    sum=a+b
    return (sum)

num1 = input("Enter first number: ")
num2 = input("Enter second number: ")
n1 = int(num1)
n2 = int(num2)
ans = add(n1,n2)
print("Sum: ",ans)
```

Code output:

```
Enter first number: 23
Enter second number: 44
Sum: 67
```

Why Use Functions in Python - Clarity

Replace multiple lines of code with a single function call, thereby making it easier to understand and maintain

Without Function

```
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
print('Hello World')
```

With Loop

```
count = 10
while count>0:
  print("Hello World")
  count-=1
```

With Loop & Function

```
def print_msg(count, msg):
  while count>0:
    print(msg)
    count-=1

#Calling the defined function
print_msg(10, 'Hello World')
```

Why Use Functions in Python - Reuse

Functions which are tried and tested can be reused within/outside the program

Code example

```
print_msg(10, 'Hello World'
)
print_msg(15, 'Hola')
```

Prints `Hello World` 10 times

Prints `Hola` 15 times

NOTE: You can call the 'print_msg' *n* number of times

Why Use Functions in Python – Division of Labor

Workload in a large project can be divided with each person working on different functions of the program

Coder 1

```
def add(n1,n2):
  return n1+n2
```

Coder 2

```
def sub(n1,n2):
  return n1-n2
```

Coder 3

```
def mul(n1,n2):
  return n1*n2
```

Coder 4

```
def div(n1,n2):
  return n1/n2
```

NOTE: This is just an example symbolizing that all the functions (sub-tasks) can be worked upon in-parallel in a team and can be later combined in the main function

Range of Function

Python `range( )` function generates a sequence of numbers in the form of a list

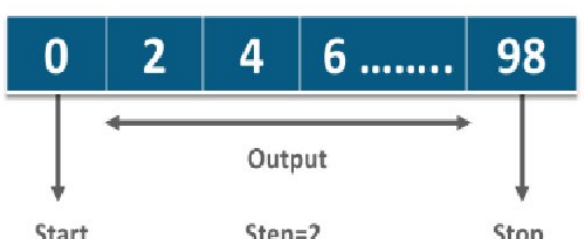
Syntax:

```
range(start,stop,step)
```

Parameter	Description
start	Integer specifying the position to start, default is 0 (Optional)
stop	Optional Integer specifying the position to end
step	Optional Integer specifying incrementation, Default is 1

Example:

```
for i in
range(0,98,2):
  print(i,end='')
```



Scope of a Variable - Function

Global Variables

Declared outside the function and can be used anywhere in the program

SCOPE

Local Variables

Declared within any function and can be used within that function only

Code example

```
a = 50
def number():
    b = 30
    print(b)

print(a)
number()
```

Code example

```
a = 50
def number():
    b = 30
    print(b)

print(a)
number()
```

Scope of a Variable – Function (contd.)

Code example

```
a = 30
def sum(b):
  c=30
  sum = b+c
  print('Addition is', sum)
sum(3)
print('Value of c', c)
```

Result

```
Addition is 33
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-53-a10954081147> in <module>()
      5   print('Addition is', sum)
      6 sum(3)
----> 7 print('Value of c', c)

NameError: name 'c' is not defined
```

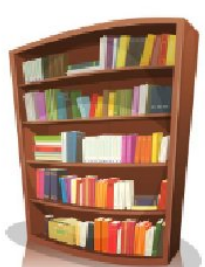
'a' is a global variable and can be accessed anywhere in the program

'c' is a local variable and so it cannot be accessed outside the function

Packages and Modules in Python

Understanding Packages and Modules

- Standard library is a collection of tools that come with Python
- A module is a file containing Python definitions and statements
- Python modules are nothing but Python files with .py extension
- A package is a collection of modules example, SciPy is a collection of modules for statistical operations



Standard Libraries

01 Built-in functions

02 Modules

03 Packages

Import Statement

- Use `import` to load a module in Python or import modulename as desiredname
- The module name is an identifier when it is imported
- When the interpreter encounters an import statement, it imports the module, if the module is present in the search path

Code example

```
import math
import random as r
```

Dir Function

A "dir" is a built-in function which returns the list of strings containing the name defined by the module

Code example

```
import math
print(dir(math))
```

Functions inside math library

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2',
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmo
d', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log1
0', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

Import Statement

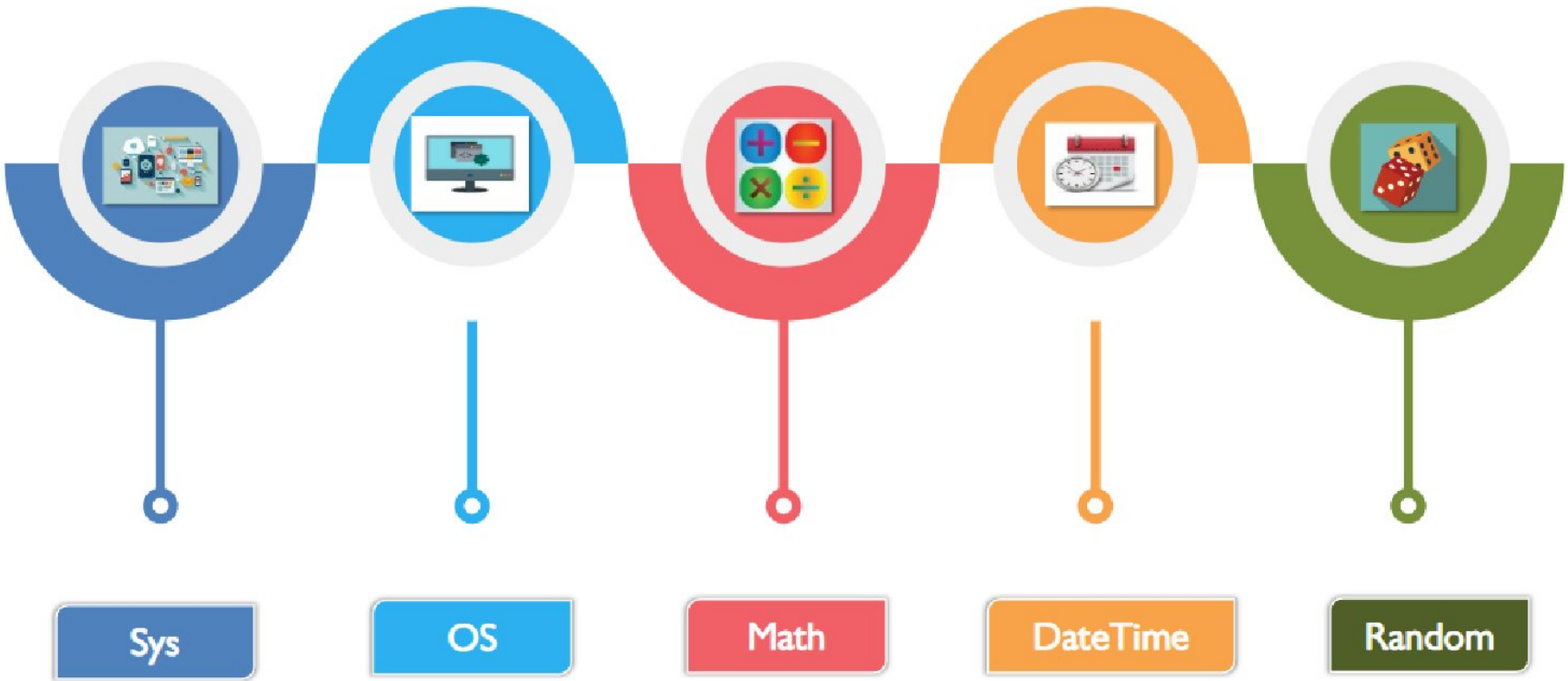
The `from import` statement does not import the entire module, it just imports required files in the module

from math import sqrt

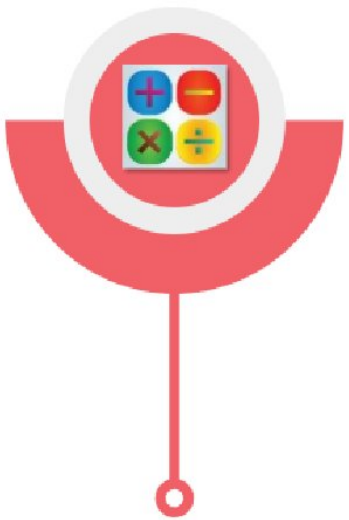
The `from import *` allows you to import all the attributes from the required module
This provides an easy way to import all the items from a module into the current namespace

from math import *

Important Modules in Python



Math Module



Math

Math module provides access to the mathematical functions

Code example

```
import math

print(math.ceil(10.098))

print(math.copysign(10,-1))

print(math.fabs(-19.7))
```

Return the ceiling of x as an integer

Return x with the sign of y. On a platform that supports signed zeros, `copysign(1.0, -0.0)` returns -1.0

Return positive integer of entered negative value

Code output

```
11
-10.0
19.7
```

Random Module

This module implements pseudo-random number generators for various distributions.

Random

Code example

```
import random

num = random.randrange(100)
print(num)

ran=random.randrange(0,100,20)
print(ran)

inte=random.randint(0,30)
print(inte)
```

Generates a random integer within the given range

Return a randomly selected element from range(start, stop, step)

Return a random integer N such that a <= N <= b

Code output

```
39
20
30
```

File Handling in Python

Opening and Closing Files

- Before reading and writing any data into a file, it is important to learn how to open and close a file
- Unless you open a file, you cannot write anything in a file or read anything from it
- And once you are done with reading or writing, you should close the file
- Here are the `open()` and `close()` functions

Opening Files

Code Example

```
file_Object=open(file_name,[access_mode])
```

Name of the file that you want to access

Mode in which the File has to be opened

Closing Files

Code Example

```
file.close()
```

Open Function – Some Access Modes

Modes	Description
r	This is the default mode: Opens a file for reading only
r+	Opens a file for both reading and writing
a	Opens a file for appending
a+	Opens a file for both appending and reading

Writing and Reading Files

`fileObject.write(string)` → The `write()` method writes content in an open file. Python strings can have binary data and not just text

`fileObject.read([count])` → The `read()` method reads a string from an open file "count" - counts number of lines in a file

Renaming and Removing Files

Renaming files

```
os.rename(current_file_name, new_file_name)
```

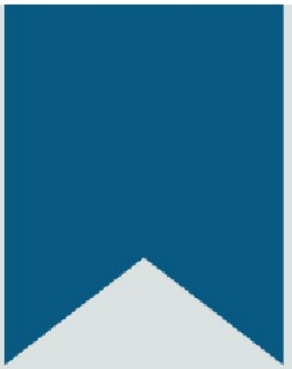
The `rename()` method takes two arguments, the current filename and the new filename

Removing files

```
os.remove(file_name)
```

The `remove()` method removes the file

Classes and Objects



Creating a Class and Object

```
class_name(object):statement(s)
```

Code Example

```
#Creating a Class
class number():
    pass

#Creating Instance of
Class
x=number()
print(x)
```

Output

```
<__main__.number object at 0x7efcf038c860>
```