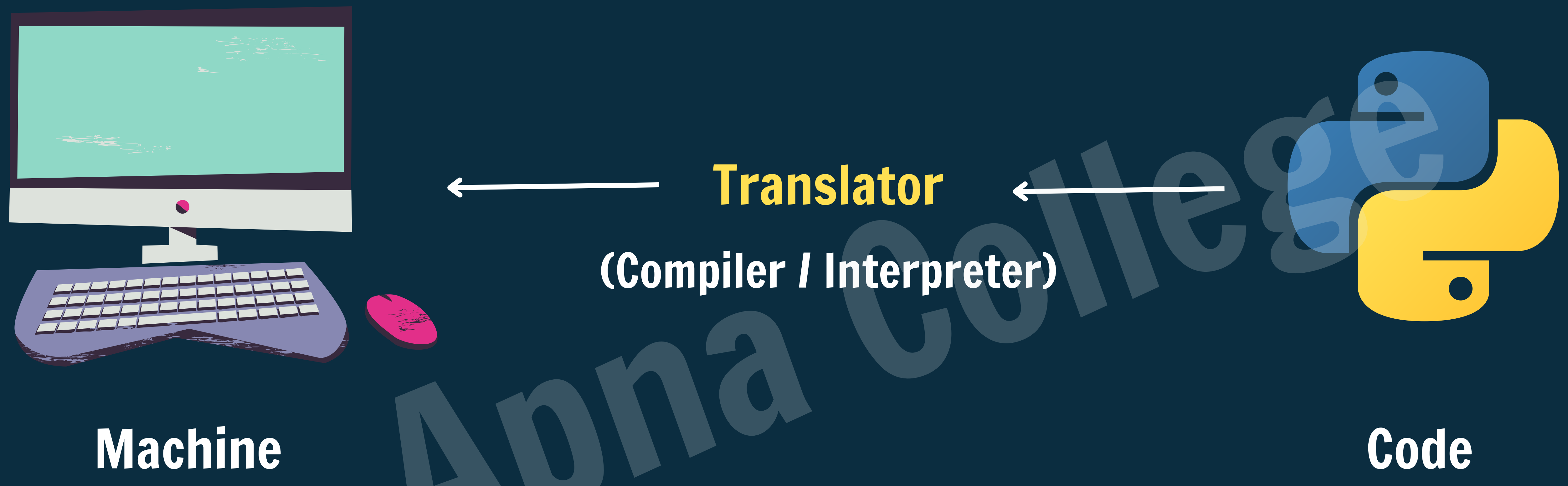
Programming



Machine

Translator

(Compiler / Interpreter)

Code

What is Python?

- Python is simple & easy
- Free & Open Source
- High Level Language
- Developed by Guido van Rossum
- Portable

Our First Program

```
print("Hello World")
```

Python Character Set

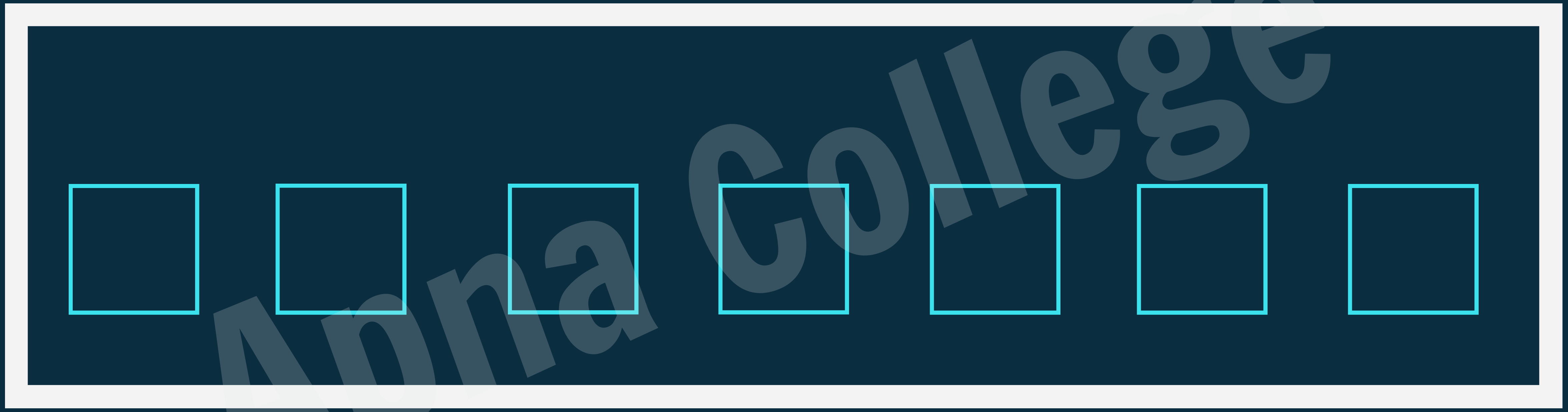
- Letters - A to Z, a to z
- Digits - 0 to 9
- Special Symbols - + - * / etc.
- Whitespaces - Blank Space, tab, carriage return, newline, formfeed
- Other characters - Python can process all ASCII and Unicode characters as part of data or literals

Variables

A variable is a name given to a memory location in a program.

```
name = "Shradha"
age = 23
price = 25.99
```

Memory

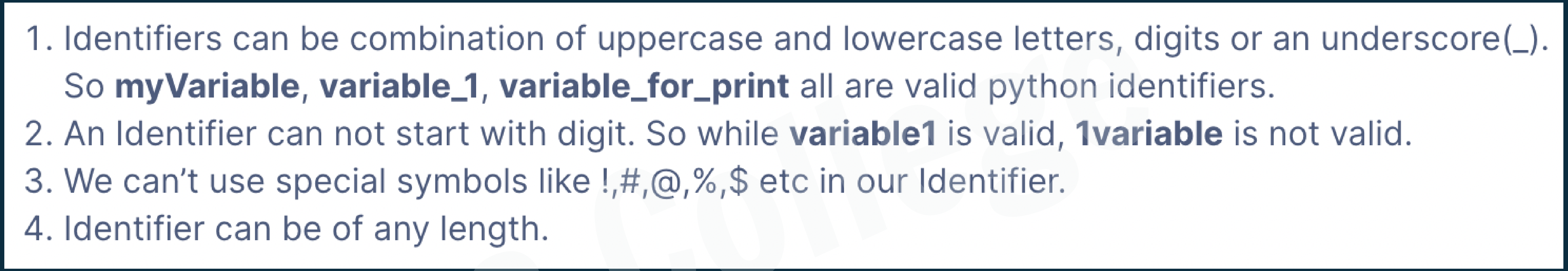


```
name = "Shradha"

age = 23

price = 25.99
```

Rules for Identifiers

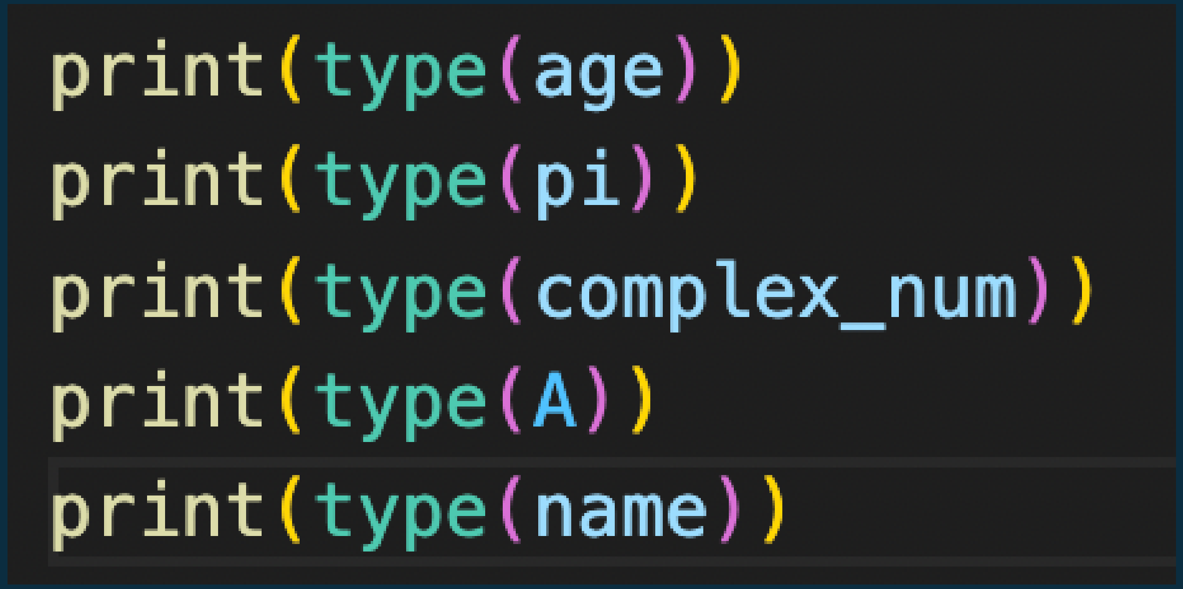


1. Identifiers can be combination of uppercase and lowercase letters, digits or an underscore(_). So **myVariable**, **variable_1**, **variable_for_print** all are valid python identifiers.
2. An Identifier can not start with digit. So while **variable1** is valid, **1variable** is not valid.
3. We can't use special symbols like !,#,@,%,$ etc in our Identifier.
4. Identifier can be of any length.

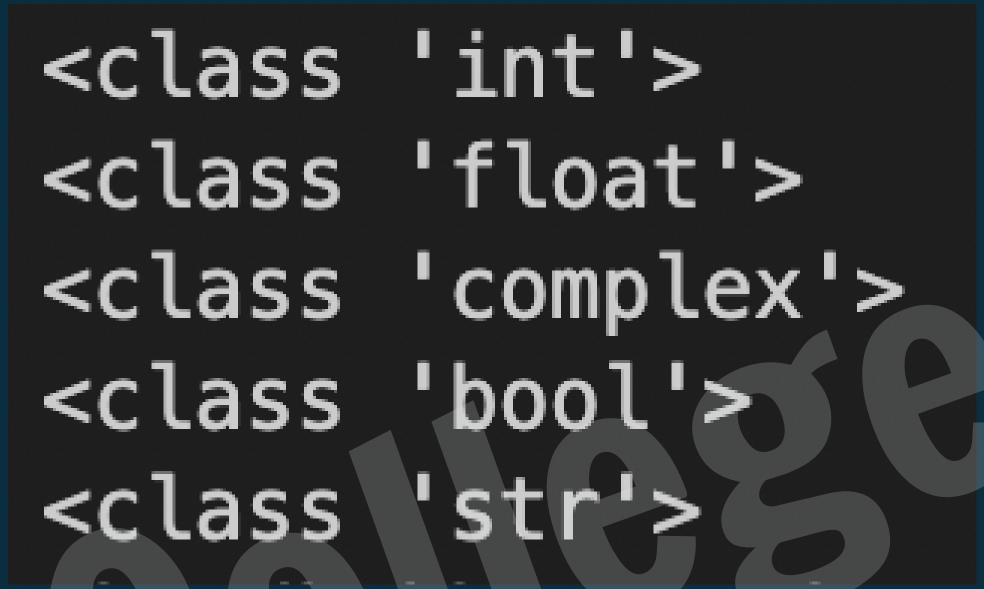
Data Types

- Integers
- String
- Float
- Boolean
- None

Data Types



```
print(type(age))
print(type(pi))
print(type(complex_num))
print(type(A))
print(type(name))
```

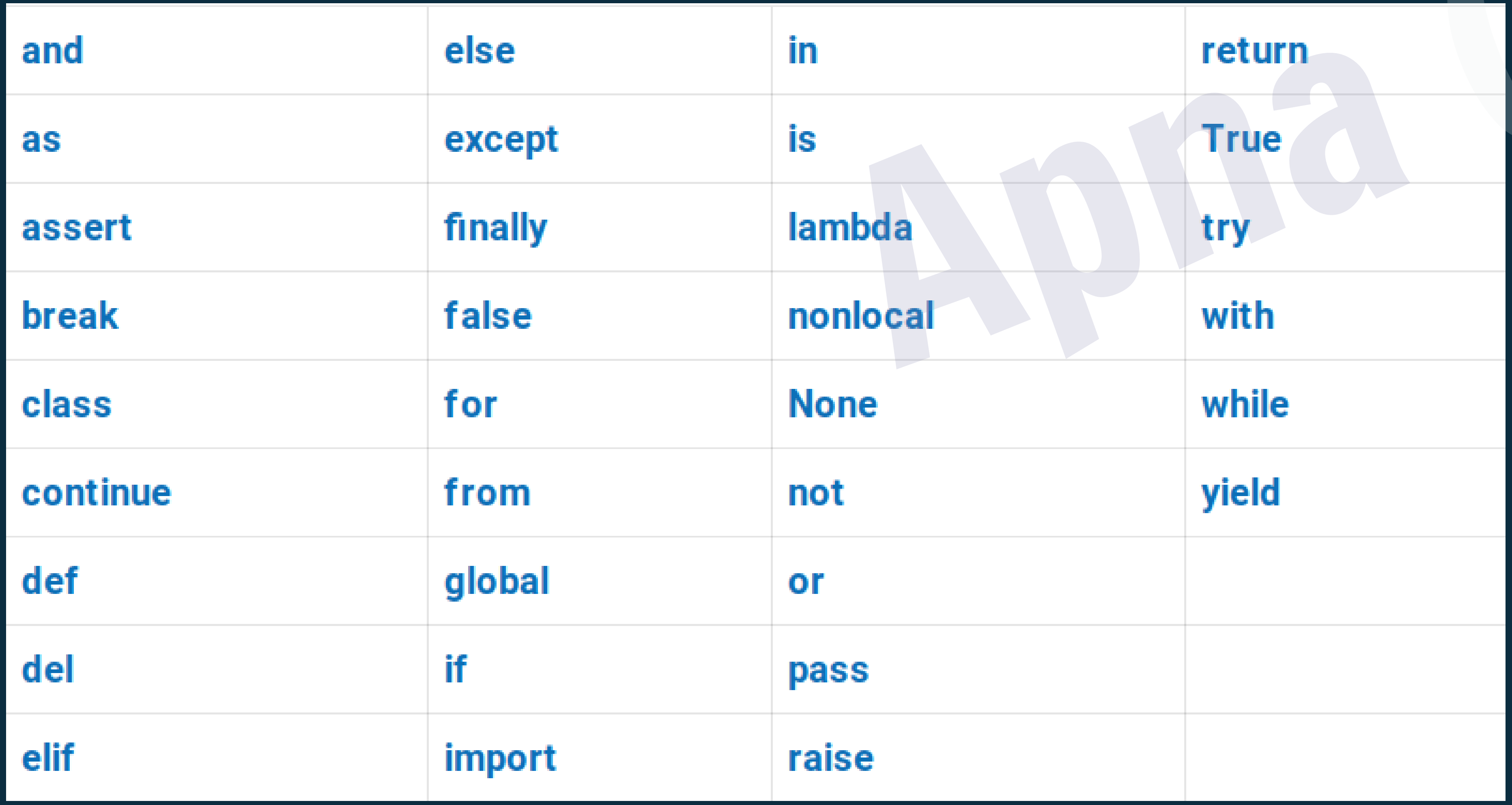


```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'bool'>
<class 'str'>
```

Keywords

Keywords are reserved words in python.

*False should be uppercase



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

Print Sum

Comments in Python

```
# Single Line Comment


"""

Multi Line

Comment

"""
```

Types of Operators

An operator is a symbol that performs a certain operation between operands.

- Arithmetic Operators (+ , - , * , / , % , **)
- Relational / Comparison Operators (== , != , > , < , >= , <=)
- Assignment Operators (= , +=, -= , *= , /= , %= , **=)
- Logical Operators (not , and , or)

Type Conversion

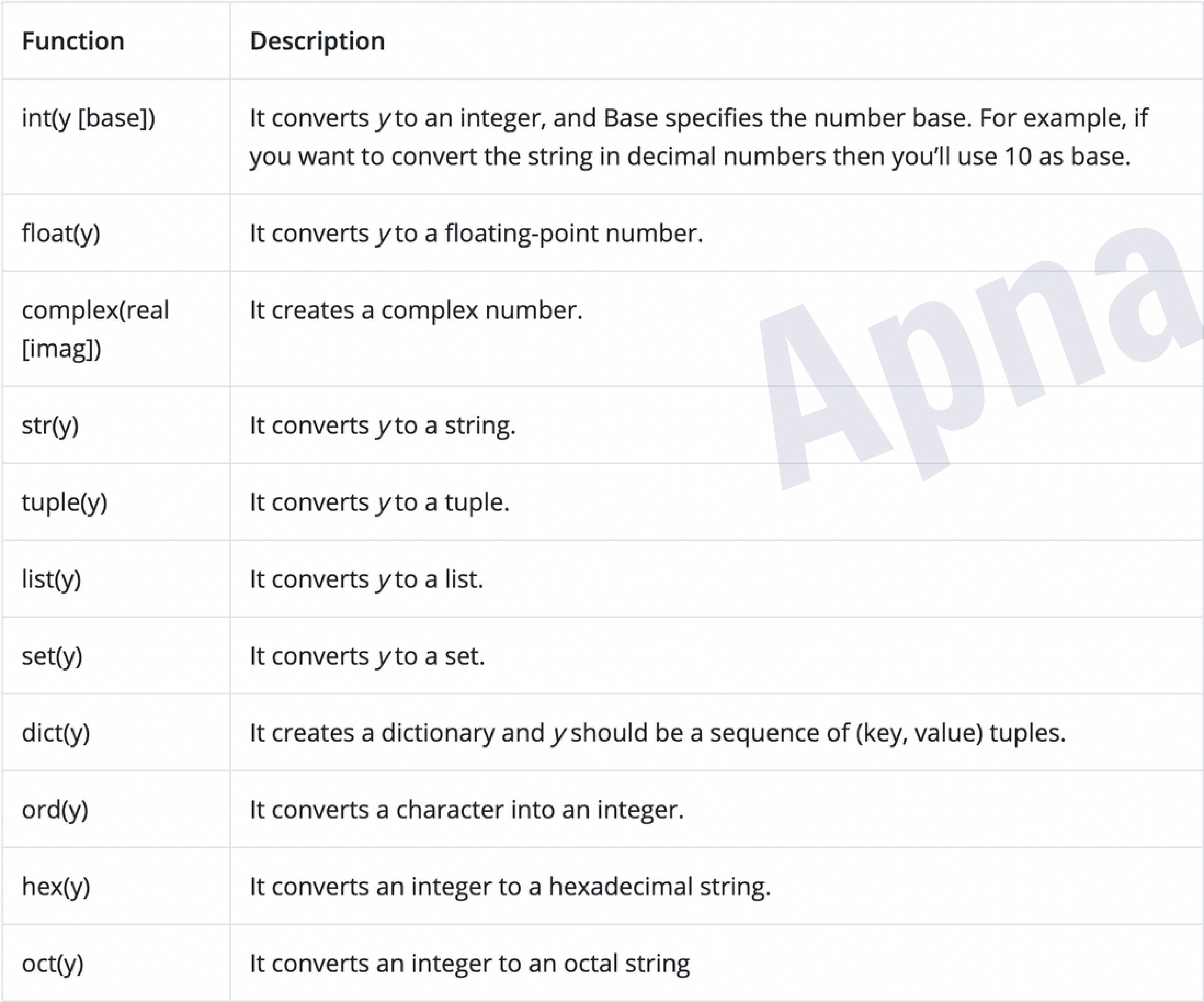
```
a, b = 1, 2.0
sum = a + b
```

```
#error
a, b = 1, "2"
sum = a + b
```

Type Casting

```
a, b = 1, "2"
c = int(b)
sum = a + c
```

Type Casting



Function	Description
int(y [base])	It converts *y* to an integer, and Base specifies the number base. For example, if you want to convert the string in decimal numbers then you'll use 10 as base.
float(y)	It converts *y* to a floating-point number.
complex(real [imag])	It creates a complex number.
str(y)	It converts *y* to a string.
tuple(y)	It converts *y* to a tuple.
list(y)	It converts *y* to a list.
set(y)	It converts *y* to a set.
dict(y)	It creates a dictionary and *y* should be a sequence of (key, value) tuples.
ord(y)	It converts a character into an integer.
hex(y)	It converts an integer to a hexadecimal string.
oct(y)	It converts an integer to an octal string

Input in Python

input( ) statement is used to accept values (using keyboard) from user

```
input( )  #result for input( ) is always a str

int ( input( ) )  #int

float ( input( ) )   #float
```

give code eg of all 3

Let‘s Practice

Write a Program to input 2 numbers & print their sum.

Let‘s Practice

WAP to input side of a square & print its area.

Let‘s Practice

WAP to input 2 floating point numbers & print their average.

Let‘s Practice

WAP to input 2 int numbers, a and b.

Print True if a is greater than or equal to b. If not print False.

Strings

String is data type that stores a sequence of characters.

Basic Operations

- **concatenation**

  "hello" + "world" ⟶ "helloworld"

- **length of str**

  len(str)

Indexing

A	p	n	a	_	C	o	l	l	e	g	e
0	1	2	3	4	5	6	7	8	9	10	11

str = “Apna_College”

str[0] is ‘A’, str[1] is ‘p’ ...

str[0] = ‘B’ #not allowed

Slicing

Accessing parts of a string

```
str[ starting_idx : ending_idx ] #ending idx is not included

str = “ApnaCollege”

str[ 1 : 4 ] is “pna”

str[  : 4 ] is same as str[ 0 : 4]

str[ 1 :  ] is same as str[ 1 : len(str) ]
```

Slicing

Negative Index

A	**p**	**p**	**l**	**e**
-5	-4	-3	-2	-1

str = “Apple”

str[-3 : -1] is “pl”

String Functions

```
str = “I am a coder.”

str.endsWith(“er.“)  #returns true if string ends with substr

str.capitalize( )  #capitalizes 1st char

str.replace( old, new )  #replaces all occurrences of old with new

str.find( word )  #returns 1st index of 1st occurrence

str.count(“am“)  #counts the occurrence of substr in string
```

Let‘s Practice

WAP to input user’s first name & print its length.

WAP to find the occurrence of ‘$’ in a String.

Conditional Statements

if-elif-else (SYNTAX)

```
if(condition) :
        Statement1
elif(condition):
        Statement2
else:
        StatementN
```

Conditional Statements

Grade students based on marks

marks >= 90, grade = “A”

90 > marks >= 80, grade = “B”

80 > marks >= 70, grade = “C”

70 > marks, grade = “D”

Let‘s Practice

WAP to check if a number entered by the user is odd or even.

WAP to find the greatest of 3 numbers entered by the user.

WAP to check if a number is a multiple of 7 or not.

Lists in Python

A built-in data type that stores set of values

It can store elements of different types (integer, float, string, etc.)

```
marks = [87, 64, 33, 95, 76]            #marks[0], marks[1]..

student = [”Karan”, 85, “Delhi”]    #student[0], student[1]..

student[0] = “Arjun”    #allowed in python

len(student)    #returns length
```

List Slicing

Similar to String Slicing

list_name[starting_idx : ending_idx] #ending idx is not included

marks = [87, 64, 33, 95, 76]

marks[1 : 4] is [64, 33, 95]

marks[: 4] is same as marks[0 : 4]

marks[1 :] is same as marks[1 : len(marks)]

marks[-3 : -1] is [33, 95]

List Methods

```
list = [2, 1, 3]

list.append(4)  #adds one element at the end    [2, 1, 3, 4]

list.sort( )  #sorts in ascending order      [1, 2, 3]

list.sort( reverse=True )  #sorts in descending order   [3, 2, 1]

list.reverse( )  #reverses list     [3, 1, 2]

list.insert( idx, el )  #insert element at index
```

List Methods

```
list = [2, 1, 3, 1]

list.remove(1)  #removes first occurrence of element    [2, 3, 1]

list.pop( idx )  #removes element at idx
```

Tuples in Python

A built-in data type that lets us create immutable sequences of values.

```
tup = (87, 64, 33, 95, 76)    #tup[0], tup[1]..

tup[0] = 43      #NOT allowed in python


tup1 = ( )

tup2 = ( 1, )

tup3 = ( 1, 2, 3 )
```

Tuple Methods

```
tup = (2, 1, 3, 1)

tup.index( el )  #returns index of first occurrence    tup.index(1) is 1

tup.count( el )  #counts total occurrences       tup.count(1) is 2
```

Let‘s Practice

WAP to ask the user to enter names of their 3 favorite movies & store them in a list.

WAP to check if a list contains a palindrome of elements. (Hint: use copy( ) method)

[1, 2, 3, 2, 1]

[1, “abc”, “abc”, 1]

Let‘s Practice

WAP to count the number of students with the “A” grade in the following tuple.

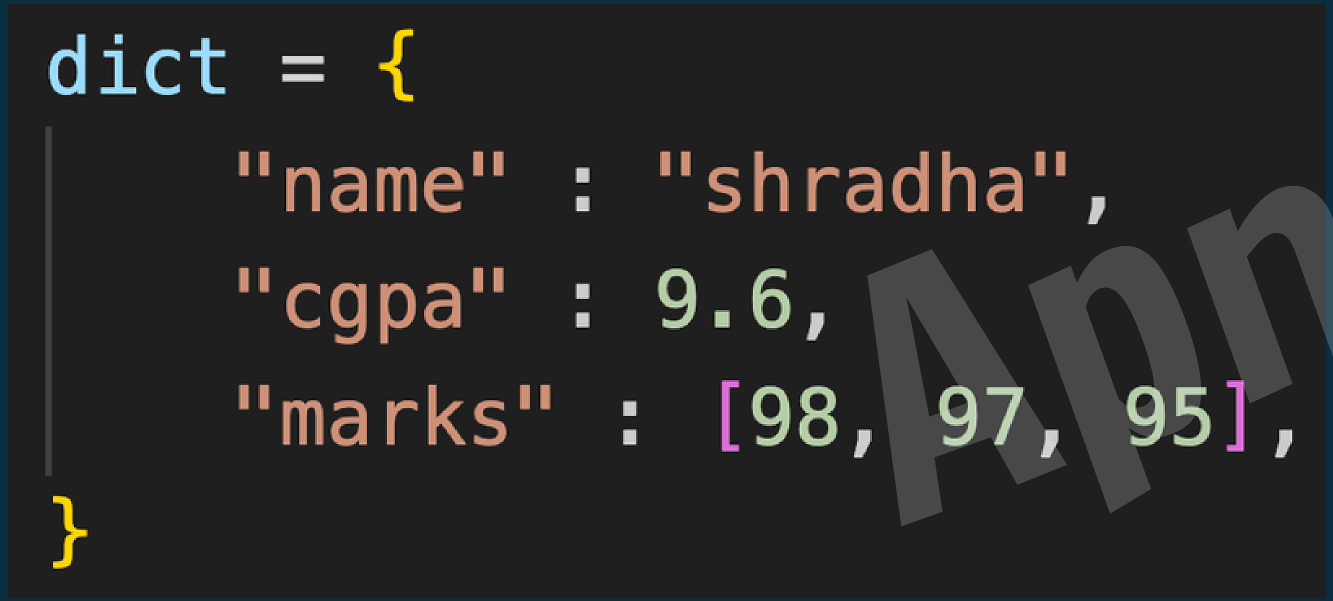
[”C”, “D”, “A”, “A”, “B”, “B”, “A”]

Store the above values in a list & sort them from “A” to “D”.

Dictionary in Python

Dictionaries are used to store data values in key:value pairs

They are unordered, mutable(changeable) & don't allow duplicate keys



```python
dict = {
    "name" : "shradha",
    "cgpa" : 9.6,
    "marks" : [98, 97, 95],
}
```

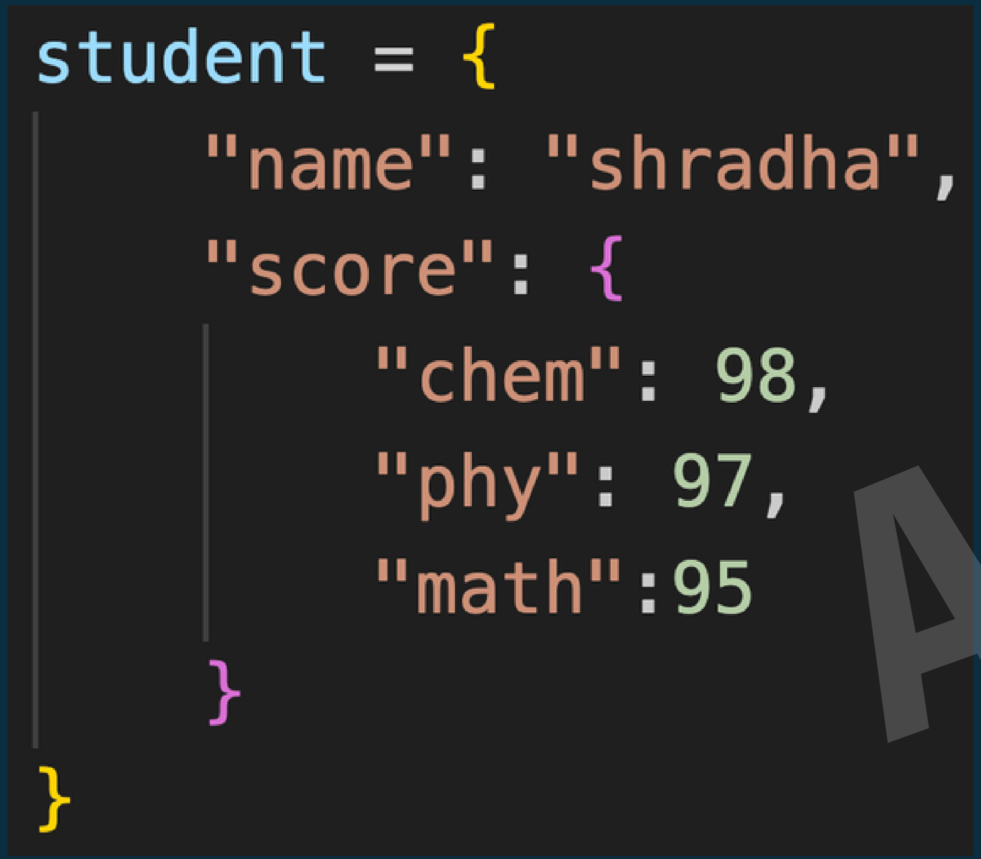
"key" : value

dict["name"], dict["cgpa"], dict["marks"]

dict["key"] = "value"   #to assign or add new

Dictionary in Python

Nested Dictionaries



```
student = {
    "name": "shradha",
    "score": {
        "chem": 98,
        "phy": 97,
        "math":95
    }
}
```

student["score"]["math"]

Dictionary Methods

```
myDict.keys( )  #returns all keys

myDict.values( )  #returns all values

myDict.items( )  #returns all (key, val) pairs as tuples

myDict.get( “key““ )  #returns the key according to value

myDict.update( newDict )  #inserts the specified items to the dictionary
```

Set in Python

Set is the collection of the unordered items.

Each element in the set must be unique & immutable.

```
nums = { 1, 2, 3, 4 }

set2 = { 1, 2, 2, 2 }
#repeated elements stored only once, so it resolved to {1, 2}


null_set = set( )        #empty set syntax
```

Set Methods

set.add(el) #adds an element

set.remove(el) #removes the elem an

set.clear( ) #empties the set

set.pop( ) #removes a random value

Set Methods

```
set.union( set2 )  #combines both set values & returns new

set.intersection( set2 )  #combines common values & returns new
```

Let‘s Practice

Store following word meanings in a python dictionary :

table : “a piece of furniture”, “list of facts & figures”

cat : “a small animal”

You are given a list of subjects for students. Assume one classroom is required for 1 subject. How many classrooms are needed by all students.

”python”, “java”, “C++”, “python”, “javascript”,
“java”, “python”, “java”, “C++”, “C”

Let's Practice

WAP to enter marks of 3 subjects from the user and store them in a dictionary. Start with an empty dictionary & add one by one. Use subject name as key & marks as value.

Figure out a way to store 9 & 9.0 as separate values in the set.
(You can take help of built-in data types)

Loops in Python

Loops are used to repeat instructions.

while Loops

```
while condition :
    #some work
```

print hello 5 times
print numbers from 1 to 5

show infinite, iterator

Let‘s Practice

Print numbers from 1 to 100.

Print numbers from 100 to 1.

Print the multiplication table of a number n.

Print the elements of the following list using a loop:

[1, 4, 9, 16, 25, 36, 49, 64, 81,100]

Search for a number x in this tuple using loop:

[1, 4, 9, 16, 25, 36, 49, 64, 81,100]

Break & Continue

Break : used to terminate the loop when encountered.

Continue : terminates execution in the current iteration & continues execution of the loop with the next iteration.

take search example
& stop the search when found

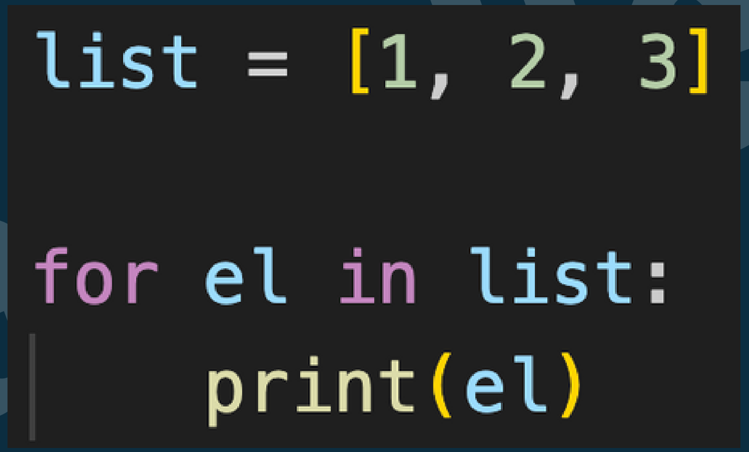
print all numbers but not multiple of 3

Loops in Python

Loops are used used for sequential traversal. For traversing list, string, tuples etc.

for Loops

```
for el in list:
    #some work
```



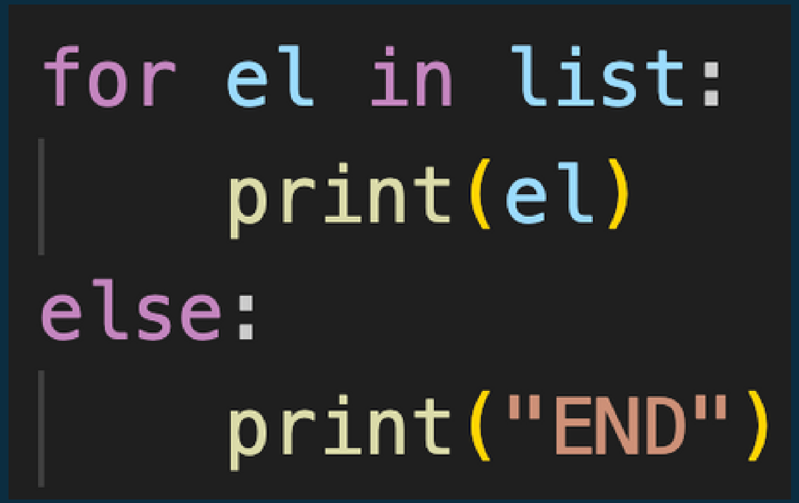
```
list = [1, 2, 3]

for el in list:
    print(el)
```

for Loop with else

```
for el in list:
    #some work

else:
    #work when loop ends
```



```
for el in list:
    print(el)
else:
    print("END")
```

else used as it doesn't execute when break is used

Let's Practice

using for

Print the elements of the following list using a loop:

[1, 4, 9, 16, 25, 36, 49, 64, 81,100]

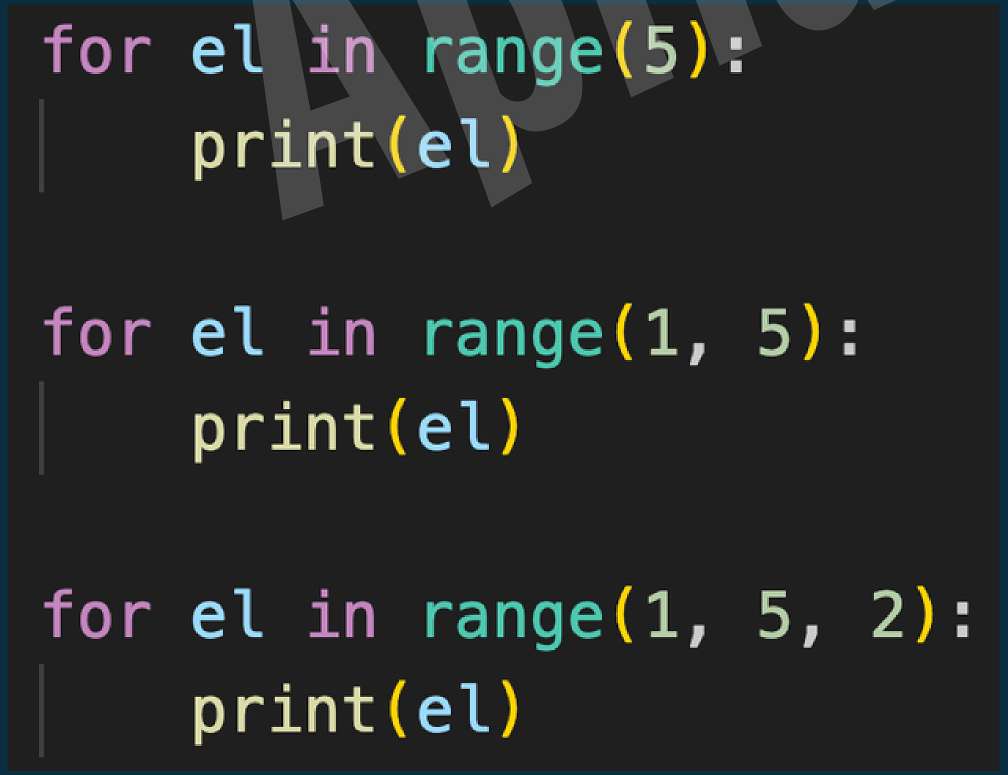
Search for a number x in this tuple using loop:

[1, 4, 9, 16, 25, 36, 49, 64, 81,100]

range( )

Range functions returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

range(start?, stop, step?)



```
for el in range(5):
    print(el)

for el in range(1, 5):
    print(el)

for el in range(1, 5, 2):
    print(el)
```

Let‘s Practice

using for & range( )

Print numbers from 1 to 100.

Print numbers from 100 to 1.

Print the multiplication table of a number n.

pass Statement

pass is a null statement that does nothing. It is used as a placeholder for future code.

```
for el in range(10):
    pass
```

generally used in execption handling

Let‘s Practice

WAP to find the sum of first n numbers. (using while)

WAP to find the factorial of first n numbers. (using for)

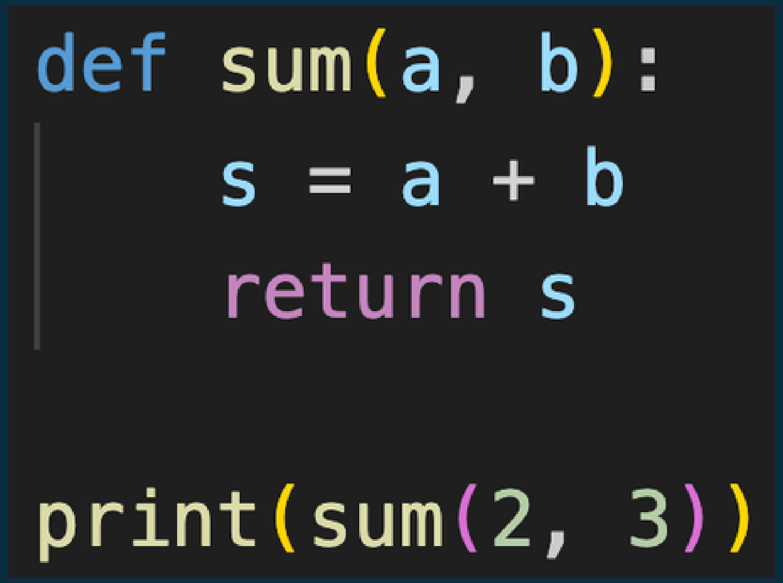
Functions in Python

Block of statements that perform a specific task.

```
def func_name( param1, param2..) :
      #some work
      return val
```

Function Definition

```
func_name( arg1, arg2 ..) #function call
```



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

print(sum(2, 3))
```

Functions in Python

Built-in Functions

print( )

len( )

type( )

range( )

User defined Functions

Default Parameters

Assigning a default value to parameter, which is used when no argument is passed.

Let‘s Practice

WAF to print the length of a list. (list is the parameter)

WAF to print the elements of a list in a single line. (list is the parameter)

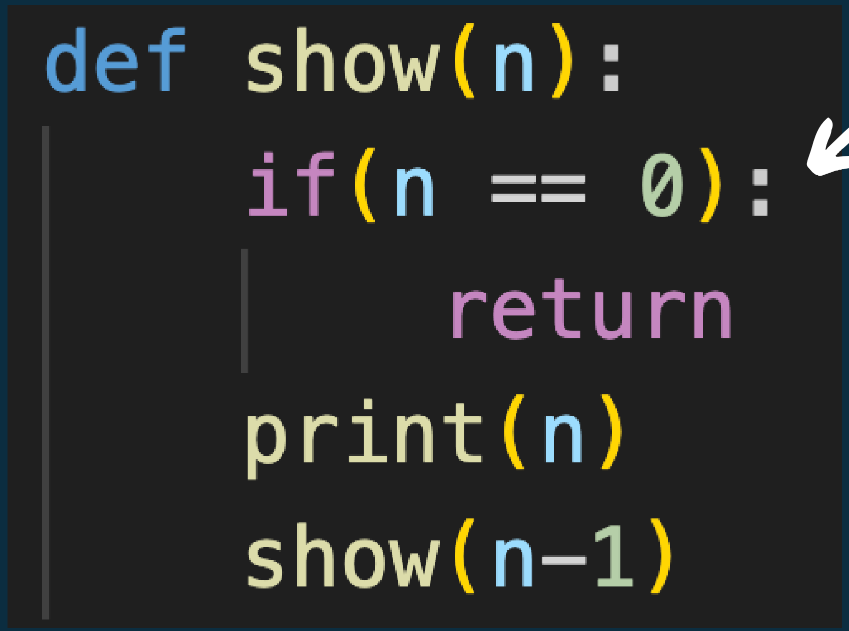
WAF to find the factorial of n. (n is the parameter)

WAF to convert USD to INR.

Recursion

When a function calls itself repeatedly.

```
#prints n to 1 backwards
```

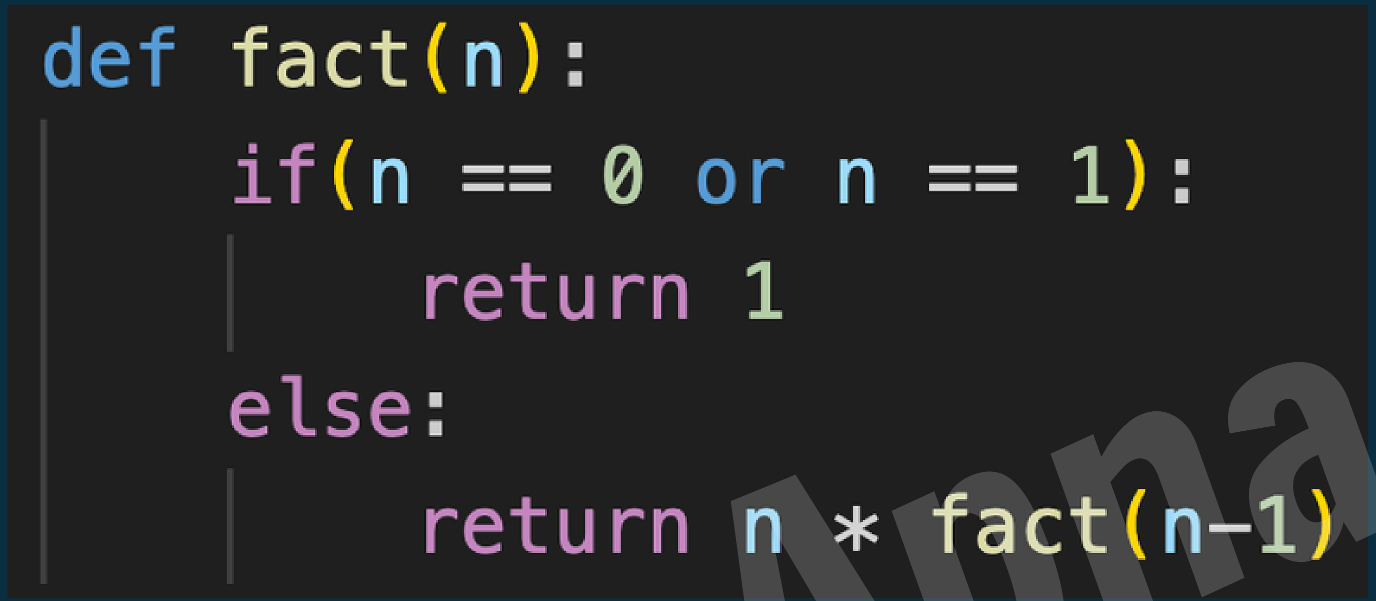


```
def show(n):
    if(n == 0):
        return
    print(n)
    show(n-1)
```

Base case

Recursion

#returns n!



```python
def fact(n):
    if(n == 0 or n == 1):
        return 1
    else:
        return n * fact(n-1)
```

Let's Practice

Write a recursive function to calculate the sum of first n natural numbers.

Write a recursive function to print all elements in a list.

Hint : use list & index as parameters.

File I/O in Python

Python can be used to perform operations on a file. (read & write data)

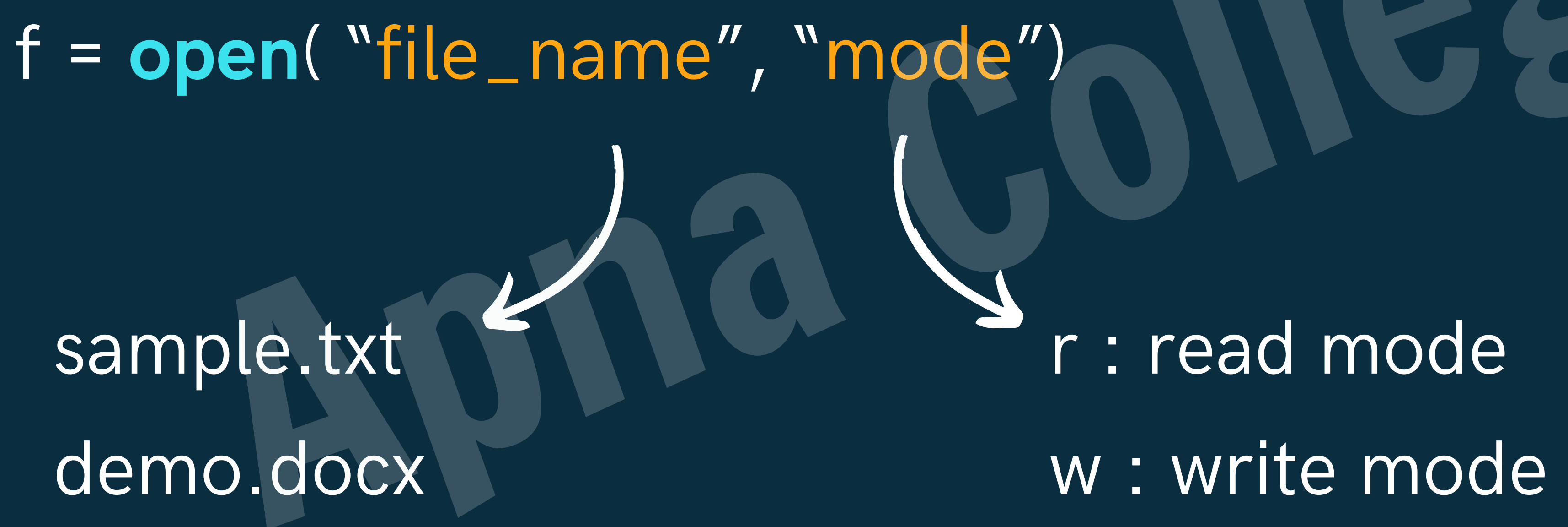
Types of all files

1. Text Files : *.txt, .docx, .log etc.*

2. Binary Files : *.mp4, .mov, .png, .jpeg etc.*

Open, read & close File

We have to open a file before reading or writing.



```
f = open( “file_name”, “mode”)
```

sample.txt

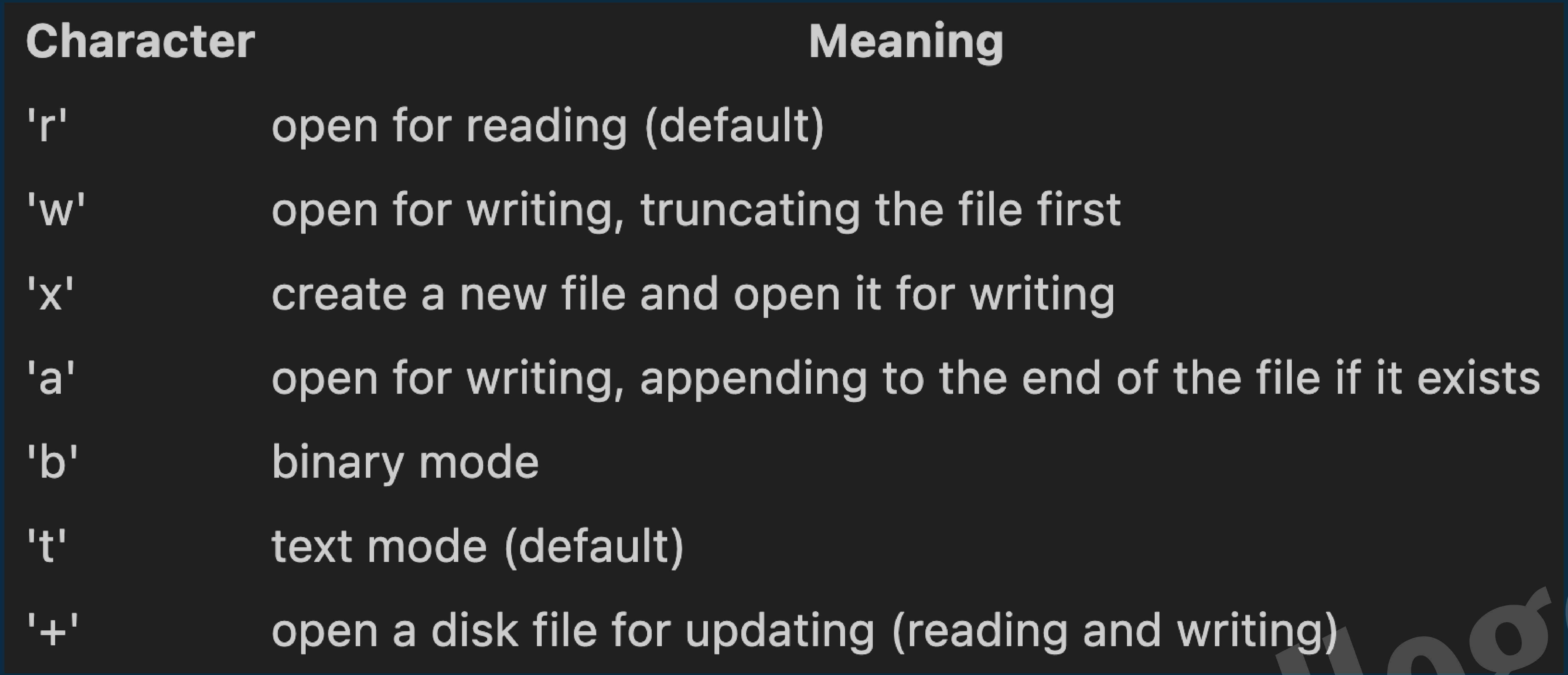
demo.docx

r : read mode

w : write mode

```
data = f.read( )

f.close( )
```



Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	create a new file and open it for writing
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)

Reading a file

```
data = f.read( )         #reads entire file

data = f.readline( )     #reads one line at a time
```

Writing to a file

```
f = open( “demo.txt”, “w”)

f.write( “this is a new line“ )    #overwrites the entire file
```

```
f = open( “demo.txt”, “a”)

f.write( “this is a new line“ )    #adds to the file
```

with Syntax

```
with open( “demo.txt”, “a”) as f:

        data = f.read( )
```

Deleting a File

using the os module

Module (like a code library) is a file written by another programmer that generally has a functions we can use.

```
import os

os.remove( filename )
```

Let‘s Practice

Create a new file “practice.txt” using python. Add the following data in it:

Hi everyone

we are learning File I/O

using Java.

I like programming in Java.

WAF that replace all occurrences of “java” with “python” in above file.

Search if the word “learning” exists in the file or not.

Let‘s Practice

WAF to find in which line of the file does the word “learning”occur first.
Print -1 if word not found.

From a file containing numbers separated by comma, print the count of even numbers.

OOP in Python

To map with real world scenarios, we started using objects in code.

This is called object oriented programming.

Class & Object in Python

Class is a blueprint for creating objects.

```
#creating class

class Student:
    name = “karan kumar”


#creating object (instance)

s1 =  Student( )
print( s1.name )
```

Class & Instance Attributes

Class.attr

obj.attr

_ _init_ _ Function

Constructor

All classes have a function called _init_(), which is always executed when the object is being initiated.

```
#creating class

class Student:
      def _ _init_ _( self, fullname ):
            self.name =  fullname
```

```
#creating object

s1 =  Student( “karan” )
print( s1.name )
```

*The **self** parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

Methods

Methods are functions that belong to objects.

```
#creating class

class Student:
    def __init__( self, fullname ):
        self.name =  fullname

    def hello( self ):
        print( “hello”, self.name)
```

```
#creating object

s1 =  Student( “karan” )
s1.hello( )
```

Let‘s Practice

Create student class that takes name & marks of 3 subjects as arguments in constructor.
Then create a method to print the average.

Static Methods

Methods that don’t use the self parameter (work at class level)

```
class Student:
    @staticmethod    #decorator
    def college( ):
        print( “ABC College” )
```

*Decorators allow us to wrap another function in order to extend the behaviour of the wrapped function, without permanently modifying it

Important

Abstraction

Hiding the implementation details of a class and only showing the essential features to the user.

Encapsulation

Wrapping data and functions into a single unit (object).

Let‘s Practice

Create Account class with 2 attributes - balance & account no.

Create methods for debit, credit & printing the balance.