PYTHON BASICS

★ General Information :-

Whitespace matters! Indent where needed.
Import modules " import modulename"

```
# This is a comment.
print("Hello World!")  # Prints to screen.
```

★ Conditional Statements :-

```
if isSunny:
    Print('It's Sunny!')
elif 90 <= temp < 100 and bath > 80:
    print('Bath is hot and full!')
elif not ((job == 'qa' or (usr == 'adm')):
    print('Match if not qa or adm')
else:
    print('No match. job is' + job)
```

ATUL KUMAR (LINKEDIN).
NOTES GALLERY (TELEGRAM).

★ Strings :-

```
title = 'Us and them'
# most list operations          Works on strings
title[0]                        # output: 'U'
len(title)                      # 11
title.split(' ')                #['us','and','them']
':'.join(['A','B','C'])         #'A':'B':'C'
nine = str(9)                   # convert int into string
title.replace('them','us')      # us and us.
```

★ Dictionaries :-

```
votes = {'red': 3 , 'blue' : 5 }.
votes.keys ( )                  # Output : ['blue', 'red']
votes['gold'] = 4               # add a key / val
del votes ['gold']              # deletes key
votes['blue'] = 6               # change value
len (votes)                     # 2
votes. values ( )               # [6,3 ]
'green' in votes                # False
votes. has_key('red')           # True
```

★ Numbers :-

```
total = 3 * 3                   # Output : 9
total = 5 + 2 * 3               # Output : 11
cost = 1.50 + 3.75              # Output : 5.25
total = int ("g") + 1           # Output : 10
```


★ Tuples :-

Like lists, except they cannot be changed

```
tuple 1 = (1, 2, 3, "a ", "z")  # Creates tuple
tuple 1 [3 ]                    # 'a'
```

★ Lists :-

```
scores = ['A', 'c', 90 , 75 , 'C ']
scores [0]                      # Output : 'A'
scores [1:3]                    # 'C', 90
scores.count ('c')              # 2
scores.append (100)             # Adds 100 to list
scores.POP ( )                  # removes third item.
```

Python OOPs Concepts

Object Oriented Programming.

Python is a multiparadigm programming language. It supports different programming approches.
One of the most popular approches to solve programming problem by creating objects. This is known as Object Oriented programming. (OOP).

- OOP has two characteristics
  1). Attributes
  2). Behavior

Example :- A parrot is an object, as it has following properties.

- name, age, color as attributes.
- Singing, dancing as behavior.

The concepts of OOPs in Python focuses on creating reusable code. This concept is also known as DPY (Don't Repeat Yourself).

Class.

A class is a blueprint for the object. We can think of class as a skech of parrot with labels. It contains all details about the name, colors, size etc.

ex.

```
Class Parrot:
    Pass.
```

Here class keyword define on empty class parrot from class we construct instances (-)n instance is a specific object created from a particular class.

Object :-

An object (instance) is an instantiation of a class. When class is defined only the discription for the class object is defined. Therefore, no memory or storage is allocated.

ex:-

```
Obj = Parrot ( )
```

Here object is an object of class parrot.
Suppose we have details of parrots. Now we are going to show how to build the class and object of parrots.
we can access the class attribute using _ class _ species.

Inheritance :-

Inheritance is a way of creating new class for using details of an existing class without modifying it. The newly formed class is derived class, simillarly, the existing class is a base class.

- ex:- Use of Inheritance in python.

```
Class Bird:
    def_init_(self):
        Print ("Bird is ready")
    def swim (self):
        Print ("swim faster")
Class Penguin (Bird):
    def-init (self):
        super ( ).-init_ ( )
    def run (self):
        Print ("Run fast")
P= Penguin ( )
P. swim ( )
P. run ( )
```

Output

Swim faster
run fast.

We can use the super() function inside the __init__() method. This allows us to run __init__() method.

Encapsulation :-

Using OOP in python, we can restrict access to methods and variables. This prevents data from direct modification which is called encapsulation. In python we donate private attribute using _ as the prefix. i.e single _ or double __

```
Class computer:
    def __init__(self):
        self.__maxprice = 900
    def setmaxprice(self, price):
        self._maxprice = price.
C = computer()
C.__maxprice = 1000
C.setMaxPrice(100)
```

We used __init__() method to store the maximum selling price of computer.

C._maxprice = 1000

Method	Variable.

Class Member access specifier	Access from own Class	Accessible from derived	Accessible from Object.
Private	Yes	No	No
Protected	yes	Yes	No
Public	Yes	Yes	No

Polymorphism :-

Polymorphism is an ability to use a common interface for multiple forms (data types) Polymorphism in python defines methods in the child class that have the same name as the method in the parent class. It is possible to modify a method in a child class that it has inherited from child class .. Parent class.

```
Class parrot :
    def fly(self):
        Print("parrot confly")
    def swim(self)
        Print("parrot can't swim")

Class penguin:
    def fly(self):
        Print("Penguin can't fly")
    def swim(self):
        Print("Penguin can swim")

def flying_test(bird):
    bird.fly()

blue = Parrot()
Peg = Penguin()

flying_test(blu).
flying_test(Peg).
```

Output

Parron can fly.
Penguin can't fly.

PYTHON FUNCTIONS

What are Functions?

A Function is a block of code only runs when is called.
In Python a function is defined using the def keyword.

```
def my_function():
    print("Hello")
```

Arguments in a Function

Arguments are specified after the function name, inside the parentheses.
You can add as many arguments as you want separate them with a comma.

```
def my_function(fname)
    Print(fname + "Refsnes")

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


Arbitary Arguments

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("soul", "god1", "txpark")
```

Keyword Arguments (Kwargs)

You can also send arguments with the key = value syntax.

```
def my_function(child3, child2, child1):
  Print("The Youngest child is" + child3)
my_function(child1 = "mcb", child2 = "bca", child3 = "btc")
```

ATUL KUMAR (LINKEDIN).

Arbitary 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.

```
def my_function(**kid):
  Print("His last name is" + kid["lname"])
my_function(fname = "Tobias", lname = "Refsnes")
```

PYTHON OPERATOR CHEATSHEET

ATUL KUMAR (LINKEDIN).
NOTES GALLERY (TELEGRAM).

Arithmetic operators:

An arithmetic operator performs mathematical operations like addition, subtraction, division and multiplication.

OPerator	Meaning	Example
+	Add two operands	>>> 5+4 9
-	Subtract two operands	>>> 6-4 2
*	Multiply two operands	>>> 2*10 20
/	divide the left oferend by right operand	>>> 5/2 2.5
%	Which will give us remindar part	>>> 5%3 2
//	which will give integer part	>>> 5//3 1
**	raised to the power	>>> 5**3 125

Comparisions operator

Operator	Description	Example
==	If two operands values are equal, then the condition becomes true.	>>> 5==3 False
!=	Values of two operands are not equal, then condition become true.	>>> 5!=3 True
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	>>> 4>3 True
<	If the value of left operand is less than the value of right operand, then condition becomes true.	>>> 6<4 False
>=	greater than equal: If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	>>> 5>=6 False
<=	less than e	>>> 4<=6 True

Logical Operators

- Logical operators is Boolean expressions such as and, or, not.
- It is just a conditional test that a result is either true or false.

Continue ⟶

Operator	Description	Example
And	it two operands are true it become true.	>>> True and True True >>> False and True False
Or	it two operands are non-zero then condition becomes true.	>>> True or false True >>> False or False false
Not	It is used to reverse the logical state of its operand.	>>> not True False >>> not not not True False.

Assignment operator:

Operator	Description	Example
=	Assigns values from right side operands to left side operand.	>>> x = 20
+=	It adds right operand to the left operand and assign the result to left operand.	>>> x += 20
-=	It subtracts right operand from the left operand and assign the result to left operand.	>>> x -= 20
*=	It multiplies right operand with the left operand and assign the result to left operand.	>>> x *= 50
/=	It divides left operand with the right operand and assign the result to left operand.	>>> x /= 50
%=	It takes modulus using two operands and assign the result to left operand.	>>> x %= 50
**=	Performs exponential (power) calculation on operators and assign value to the left operand.	>>> x **= 50
//=	It performs floor division on operators and assign value to the left operand.	>>> x //= 50

Membership Operator

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	>>>x = "Python" >>>'p' in x True.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	>>>x = "Python" >>>'z' in x false.

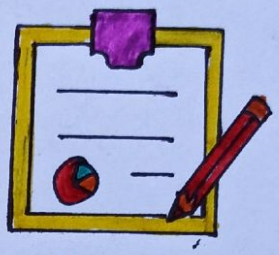
Bitwise Operators.

Operator	Description	Example
&	Operator copies a bit to the result if it exists in both operands	>>>10 &20 0
|	It copies a bit if it exists in either operand.	>>> 10|20 30
^	It copies the bit if it is set in one operand but not both.	>>> 10^20 30
~	It is unary and has the effect of 'flipping' bits.	>>>~20 -21
<<	The left operands value is moved left by the number of bits specified by the right operand.	>>>10<<2 40
>>	The left operands value is moved right by the number of bits specified by the right operand.	>>>10>>2 2

Identity Operators

Operator	Description	Example.
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	>>>x = "hello" >>>y = "hello" >>>x is y True.
is not	Evaluates to false if the variables on either side of the operator point to the same object and True otherwise.	>>>x = "hello" >>>y = "hello" >>>x is not y False.

PYTHON NOTES
VARIABLES & DATATYPE



Python Variables :

Variable is a name that is used to refer to memory location. Python is also known as an identifier and used to hold value.

In python, we don't need to specify the type of variable python is a infer and smart enough to get variable type.

example : x = 10
name = "Atul Kumar"
value = 12.2

identifier:

Variable are the example of identifier. An identifier is used to identify the literals used in the program.

Rules to name an identifier —

- The first character of the variable must be alphabet or underscore (_).
- All character except first character may be an alphabe (A-z), (a-z) or digit (0-9).

Continue →

- Identifier name must not contain any white space or special character (!, @, #, %.)

- Identifier name not similar to any keyword defined in language.

Valid vs invalid identifier

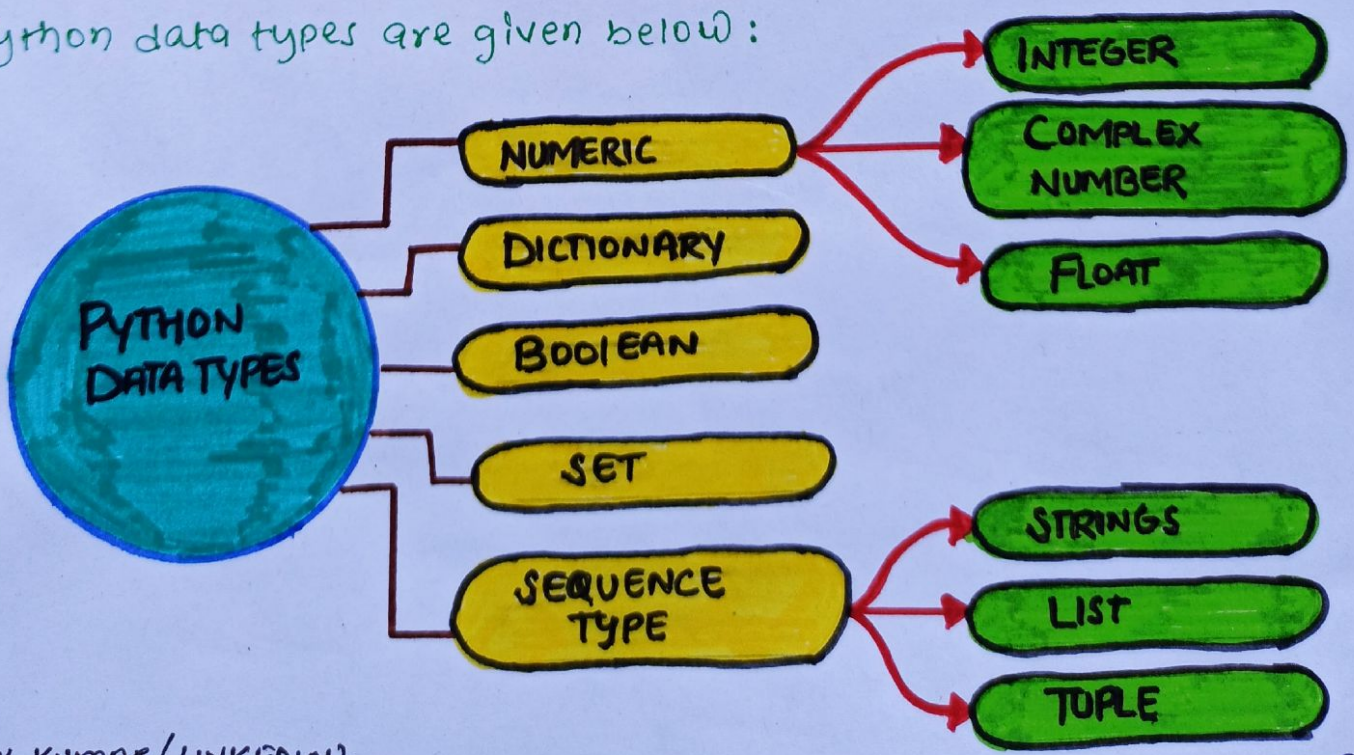
Valid	Invalid
• abc123	• 123abc
• abc_de	• abc@
• _abc	• 123
• ABC	• for
• abc	

Python Datatype

A variable can hold different type of value. for example name is must be stored in string where as id is stored in integer.

Variable can store data of different types, and different types can different thing.

Python data types are given below:



Operators in Python

The operator can be defined as a symbol which is responsible for particular operation between two operand.

Operator are used to perform operations on variable and value.

Python divides operator in following Groups:

- Arithmetic operator
- Comparison operator
- identity operator
- Membership operator.
- Assignment operator.
- logical operator
- Bitwise operator

Arithmetic Operator

Arithmetic operators are used to perform common mathematical operation.

Operator	name	example
+	addition	x + y.
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modul	x % y
* *	exponentiation	x ** y
//	floor division	x // y

Assignment Operator

assignment operator are used to assign value to variable.

Operator	example	same as
=	x = 5	x = 5
+=	x += 1	x = x + 1
-=	x -= 1	x = x - 1
*=	x *= 1	x = x * 1
/=	x /= 1	x = x / 1
%=	x %= 1	x = x % 1
//=	x //= 1	x = x // 1
**=	x **= 1	x = x ** 1
&=	x &= 1	x = x & 1
|=	x |= 1	x = x | 1

Comparison Operator

Operator	Description	example
==	equal	x == y
!=	not equal	x != y
>	greater than	x > y
<	less than	x < y
>=	Greater/ equal	x >= y
<=	less/ equal	x <= y

logical Operator

Operator	Description	example
and	Returns true if both statement are true	x<5 and x<10
Or	Returns true if one of statement is true	x<5 or x<4
not	Reverse the result, Returns false if the result is true.	not(x>5 and x>10)

Python identity Operator

Identity Operator are used to compare the object, not if they are equal. but if they are actually same object with the same memory location.

Operator	Description	example
is	Return True if both variables are the same object.	x is y
is not	Returns true if both are not same object.	x is not y.

Python membership Operator.

Membership operator are used to test if a sequence is pretended is an object.

Operator	Description	example
in	Return True if a sequence with the specified value is present.	x in y
not in	Return True if a sequance with the specified value is not present.	x not in y.

Python Bitwise operator

Bitwise operator perform bit by bit operation on the values of two operand. Consider the following table.

Continue ⟶

→

Operator	name	Description
&	AND	Set each bit to 1 if both side are 1
|	OR	Set each bit to 1 if one of two bit is 1.
^	XOR	Set each bit to 1 if only one of two bits is 1.
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off.
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off.

Input in Python

python input() function

Python input function is used to get input from user. it prompts for the user input and read a line. After reading data. it converts it into a string and returns that. it throws an error E of error is EOF is read.

Parameter

prompt : it is a string message with prompt for user input.

Return

it returns user input after converting into string.

let see an example of Input() function to understand its functionality.

Continue →

example:

Input

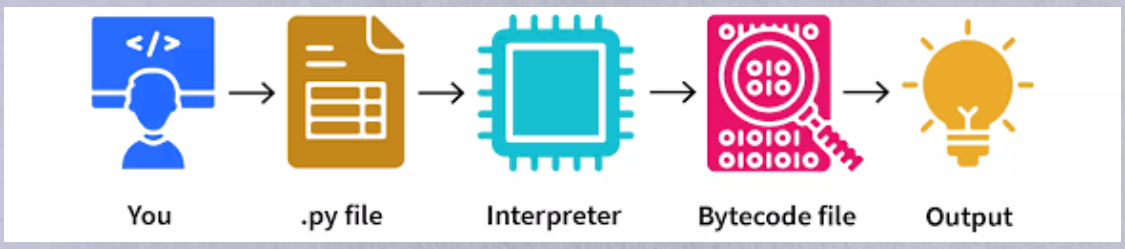
```
# Python Input() function.
val = Input("enter a value :")
print ("you enrered :", val)
```

Output

```
enter a value : 20
you entered : 20
```

1. What is Python?

→ • Python was created by Guido van Rossum, in 1991.
• Web application are made by python with the help of servers.
• Python handles huge amount of data & perform mathematics.
• Python works on different platform like (Mac, Linux, Windows, Raspberry pi etc).
• Python is a object oriented Programming language.
• Python is high level & most popular programming language.



2. Is the python case sensitive language.

→ Yes, Python is a case sensitive language. Because it differentiates the lower case & upper case identifiers.

3. What is Lambda function in Python?

→ Lambda function in python having no name. Lambda forms are utilized to construct anonymous functions.
A normal functions are defined by lambda keyword.

example :-

```
adder = lambda x, y : x + y
Print (adder (4, 8))
```

4. What are the advantages of Python?

→ • Python is very flexible & extensible language.
• Python is object oriented programming language.
• In Python there are Tuple, list and dictionary are useful data structure is present in the language.

- Python runs on various platform like Mac, Window, Linux, raspberry pi etc.
- Python is free and open source language.

5. What is PEP 8?

→ • In Python PEP 8 has appears as the style guide that most projects hold to it.
- It provides a very readable and eye-pleasing code style.
- A full form of PEP is Python Enhancement Proposal.
- PEP 8 recommends use 4 space to show Indentation and tabs should only be used to maintain consistancy in the program.

6. What is self in Python?

→ • By using self keyword we can access the attribute & methods of class in python.
- The self keyword is used to represent the instance of the class.
- The self variable in the init method refer to the newly created object while in other methods.
- Self is also used to refer to a variable field within the class.

7. What is Init function in Python?

→ • _init_ is one of the reserved method in Python.
- The _init_ method can be called when an object is created from the class.
- All class have a function called _init_( ). which is always executed when the class is being initiated.
- Use the _init_( ) function to assign values to object properties.
- The _init_ method doesn't created the object but only initialize the object's attributes.

8. What is Pass in Python?

→ • The Pass keyword represent a null operation in Python.
• It is generally used for the purpose of filling up empty block of code which may execute during runtime.
• The pass statement is generally used as a placeholder.
• The pass statement is useful when you don't write implement of function.

9. What is slice function in Python?

→ • Python slice( ) function is used to get a slice of elements from collection of elements.
• slice ( ) function used with string, list, tuple, set, bytes or range objects.
• Syntax :- slice (stop)
slice (start, stop, step)

10. How is Memory managed in Python?

→ • Memory management in python involves a private heap containing all python objects & data structure.
• The Python memory manager is deals with storage management aspects, like sharing, segmentation and caching.
• The core API gives access to some tools for programmer to code.
• In python there is also garbage collection is present. It recycles all the unused memory and so it makes free space in memory.

11. What is tuple in Python?

→ • A tuple is a built in data type.
• We can stores a value in tuple.
• A tuple in python is similar to a list.
• Tuples are immutable so we cannot change its values.
• Tuples respond to the + and * operators much like strings.

12. What is an operator in Python?

→ • Operator is a symbol which is used on values to produce output on it.
• Arithematic operators are used to performing mathematical operations like addition, subtract, multiplication & division.
• Comparison operator compares the values and returns true or false values on condition.
• Logical operator perform Logical AND, Logical OR and Logical NOT operations. It is used to combine conditional statements.
• Bitwise operator act on bit & perform bit-by-bit operations.
• Assignment operators used to assigning values to the variables.

13. Is the python interpreted language?

→ • A Python is an interpreted language.
• Python language Program runs directly from the code.
• Python converts the source code into language code. And it is again translated into machine language.
• An interpreted language is any programming language that isn't already in machine code prior to runtime.
• Python is fall under byte code interpreted.
• The .Py code source code is first compiled to byte code as .PyC

14. What are Python iterators.

→ • The iterators are objects which can be travesed through or iterated upon.
• Iterator contains countable number of values.
• Python iterator used to iterate object like list, tuple, and sets.
• The iterator object initialized using iter( ) method.

15. What is namespace in Python?

→ A namespace is naming system used to make sure that names are unique to avoid naming conflicts.

16. Is Indentation required in Python?

→ • Indentation is most important for python.
• It specifies block of code. All code with in loops classes functions etc.
• It is usually done using four space character.
• If your code is not intended necessarily.
• It will not execute accurately and will through error as well.

17. What are local and global variable in Python?

→ • There are two types of variables local and global variables.

Local Variables:-

• Local variables can only be reached within their scope.
• Any variable declared inside a function is known as a local variable. This variable is present in the local and not in the global space.

Global variables:-

• A Global variable can be used anywhere in the Program as its scope is entire program.
• Variable declared outside a function or in global space are called global variables.
• These variables can be accessed by any function in the program.