Python Programming

Nomi Intelligence

Intro

Introduction To Python

Python is a high level Programming Language created by Guido Rossum in 1989 which is very easy to read and write like simple English sentence which makes code lot easier and understandable even for new users who never interact with python Programming

- **Free & Open Source:**

Python programming code's,libraries, documentations, everything is open source means anyone can use it and makes changes according to their needs.

- **Easy To Debug:**

Python use interpreter instead of Compiler, Means the code are run line by Line if there is any error it will show you The exact line where you have error Therefore debugging is lot easier as compare to other programming languages.

- **Portable:**

you can run python code on any operating system you for example want if any python code is written in Window that could be also work in Mac, IOS or Linux Operating system.

Installation:

you can download python from https://www.python.org/downloads/ according to your operating system your using but you must download the version above than 3.5 to follow the course if you still stuck how to download then watch the video link bellow link

IDE Used:

you can use any of your ide (integrated devolopement Enviroment) which can support python3 but I will prefer to use jupyter notebook or visual studio code which are being used in this course.

First Python Program

You can write your first python program in just one line of code Like:

```
print('hello world')
```

Here's in the above Code the python print function tells the computer to display On the screen whatever inside the parenthesis of print function in the above case it will print hello world in the screen without quotation marks as we always include string or character's inside quotes will be discussed later on

Comment's

Comment's in python are such code which are ignored by the python interpreter mean's that it is just for human understanding that what code he/write if they check their code 3 or 4 months later they still can understand their codes by that comments, following are the examples:

Types Of Comments:

There are two types of comments

1. One line comments:

```
# this is my first comment
print('hello world')
```

One line comment is written by # symbol and is used if you have to comment only one line of code as in the above example where comment Give us the description of bellow code.

2. Multiple line comments:

```
'''This is multi
lines of
comments and will
not effect the
program'''
```

Multi line comment's are written between triple single quotes which is used if you want to write large number of lines as a Comment in your code as like in the above example this code will not effect Your program.

Variables

variable is like a container where we stored our values. for example: name = 'nomi' In the above example var is our variable and 'nomi' is the value which is stored inside that variable
following are some example's of variables

```
age = 25
message = 'Nomi Intelligence'
score = 44.5
```

Rules For Variable Declaration:

1. Variable Can only accept digets, alphabets and underscores
2. Variable name cannot accept spaces between any character
3. Variable name should not start from integer number's it should always start from any character or underscore
   Following are some examples of variable names

```
name = 'nomi'
youtube_channel = 'Nomi Intelligence'
Age = 24
WhatsApp_Number = +923483282236
```

Note:
python is a case sensitive language **Name** and **name** are two different Types of variables.

Operators

Following are certain types of operators in python

1. **Arithimetic Operators:**

➔ **Addition:**

Here's how we can add two values or variables together in python

```
a = 33
b = 44
print(a+b)
# or you can do it as
c = a+b
print(c)
```

➔ **Subtraction:**

In the same way you can also do subtraction

```
a = 33
b = 44
print(a-b)
```

➔ **Multiplication:**

as in normal mathimathics we use 'x' for multiply two values together but in programming we always use * for multiplication purpuse as in the following code:

```
a = 33
b = 44
print(a*b)
```

➔ **Division:**

For division we use '/' or '//' slashes if your using single slash it will show you the complete result with decemal points while if your using double slashes it will show you only a single value without decimal points

```
# use single slash
a = 334
b = 44
print(a/b)
```

```
# use double slash
a = 334
b = 44
print(a//b)
```

Assignment Operators

Assignment operators are used to assign values to each other as following are certein types of assigment operators

1. =
2. +=
3. -=

let's use it

```
# if you have to assign any variable or value to each other:
a = 2
print(a)
# if you have to use add operator
a += 2
print(a)
# if you have to use minus operator
a -= 2
print(a)
```

Comparison Operators:

Comparison Operators are being used to compare one variable or value with other's. Comparison Operators will always give's you answer in two states True & False

if the condition is true it's show you True Otherwise False Here's some of the comparison operators:

➔ == (known as equal to)
➔ != (known as not equal to)
➔ < (known as less than)
➔ > (known as greater than)
➔ <= (known as less than equal to)
➔ >= (known as greater than equal to)

Let's Use these all in the next page example:

```
a = 23
b = 45
# check weather these two variables are equal or not
print(a==b)
# check weather these two variables not equal to each other
print(a!=b)
# check weather a is less than b or not
print(a<b)
# check weather a is greater than b or not
print(a>b)
# check weather a is less than or equal to  b or not
print(a<=b)
# check wetaher a is greater or equal to b or not
print(a>b)
```

Logical Operators:

Logical operators are:

and, or, not operators.

- **And Operator:** if both statements are True it will give you the result as True that's known as AND operator
- **Or Operator:** if atleast one statement is true are known as OR operator
- **Not Operator:** it will give you True if both of the statement's become false.

Examples:

```
# Example 1
print(True == True)
print(True == False)
print(False == False)
print(True and True)
print(True or False)
```

```
# Example 2
Value1 = True
Value2 = False
print("The comparison of Value1 and Value2 is",(Value1 and Value2))
print("The comparison of Value1 or Value2 is", (Value1 or Value2))
print("Value2 is",(not Value2))
```

Special Operators

Python also provide some special operators like:

1. Identity operators

Two types of identity operators

➔**is** ==> if both values are identical to each other it will return True if not then False

➔ **is not** ==> If both values are not identical to each other it will return True otherwise False

Following are some Examples:

```
# Example
a = 23
b = 45
c = 'nomi'
d = 'sonu'
print(a is b)
print(a is not b)
print(c is d)
print(c is not d)
```

2. Membership operators

Two Types

1. in

2. not in

in check the value in a sequence if that is found it's show's True if not then False

not in check value in a sequence if that is not found it's show's True otherwise False

```
# Example
names = 'nomi', 'rizwan', 'marwan'
print('nomi' in names)
print('nomi' not in names)
```

Data Types

In python each value has a data type
Primarly there are following data types in python programming
==> **Integers:**
all the numbers without a decimal point can be consider as integer in python such as 3, 2, 400, 1000,
==> **Floating point:**
All the numbers which having decimal point in between values such as 33.44, 55.4, 0.3 etc
==> **Strings:** All the characters in python can be known as string type such as 'nomi intelligence'
==> **Boleans:** Boleans can be two types True or False

```
# Example of Integers
integers = 44
# Example of Floating Point
floating_point = 44.55
# Example of Strings
strings = 'Nomi Intelligence'
# Example of Booleans
booleans = 45.44
```

Check Type of Any Variable
If you have to check the type of any variable then you can do that as:

```
# Check type of a and b
a = 33
b = 'nomi intelligence'
print(type(a))
print(type(b))
```

If you have to take some input from the user through keyboard you can use this input function which takes input from the keyboard as a string. ==> it is a built in function which bring flexibitly to our program, for now we will only know how to take some data from the user as a string and store it in a variable. later on you will play alot with this function so then you will have full understanding about this.

```
# Example
name = input('my name is ')
```

TypeCasting

As above input function only take strings from keyboard how if we have to get int or float values from users, if you type int or float from keyboard that will be considered as string so you have to do type casting means you have to convert one type into another see the following examples

```
# Example
age = input('Enter your age ')
print(type(age))
```

As you see in the above example we type int value but the type shows that it's an integer value so we have to type cast it as follow:

```
# Example 2 type cast to int
age = int(input('Enter your age '))
print(type(age))
# if you have to type cast it to float
age = float(input('Enter your age '))
print(type(age))
```

```
# Example 3
name = input('enter your name \n')
age = input('enter your age \n')
marks = float(input('your marks is \n'))
```

Keywords

==> keywords are reserved words in python they cannot be define as variable names.
==> Keywords are case sensitive you have to write the exact define in python programming
==> All keywords are lower case except True, False & None.
==> following are some keywords in python which can be used later on in the course
keywords:
False, break, for, not, None, class, from, or, True, continue, global, pass, def, if, as, elif,in, try, etc

Strings

Strings is python data type which can be define as sequence of characters which are enclosed between quotes.
python string can be quoted in three ways:
==> single quote string can be like: 'nomi'
==> double quote string can be like: '' this is nomi's youtube '' here we can type single quote between double quotes
==> triple single quotes can be like: ''' Hi! its nomi
Welcome to everyone here '''‘

```
# Example Single Quotes
name = 'nomi'
print(name)
```

```
# Example double Quotes
name = " nomi khan's "
print(name)
```

```
# Example triple single Quotes
var =  '''this is nomi khan
Enjoy this masterpiece tutorial
from Nomi Intelligence'''
print(var)
```

String Functions:

some of the most string functions or methods to perform operations on strings are:

1.Count Function:

This function will count the character you enter in count func in your entire string characters

```
# Example:
a = 'Nomi Intelligence'
print(a.count('n'))
```

2.Len Function:¶

This function will give you the length of your total character stored in a variable

```
# Example
a = 'Nomi Intelligence'
print(len(a))
```

3.Replace Function

This function will replace any character with another character in your orignal sequence

```
# Example
a = 'Nomi Intelligence'
print(a.replace('Nomi','khan'))
```

The above code will replace 'Nomi' with 'khan' in your orignal variable

4.Find Function:

This function will give you the first occurence index of the character you wanna get

```
# Example
a = 'Nomi Intelligence'
print(a.find('m'))
```

This will give you index 2 as character 'm' is availaible at index 2.

5.EndsWith Function

This function will give you the answer in True or False that your exact string ends with the character you enter or not.

```
# Example
a = 'Nomi Intelligence'
print(a.endswith('ce'))
```

6.Capitalize Function:

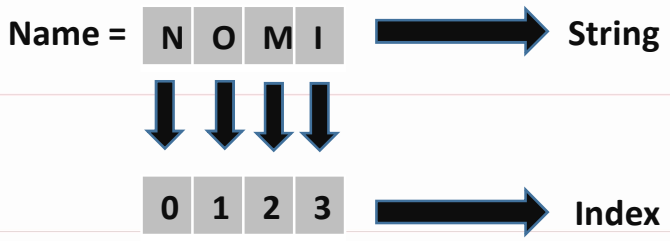
This function will capatalize the first character in your sequence of strings.

```
# Example
a = 'nomi intelligence'
print(a.capitalize())
```

String Slicing:

As you know that index always start from 0,1,2,3... and so on what if you have to find a character in your entire string, you can do that through slicing or indexing.

Example:



Following are the practical examples:

```
# Find characters from 0 to 3
a = 'Nomi'
print(a[0:3])
```

```
# Find characters from 0 to 3
a = 'Nomi'
print(a[0:4])
```

```
# Find character at second index
a = 'Nomi'
print(a[2])
```

```
# Find all the characters
a = 'Nomi'
print(a[:])
```

Negative Indexing:

You can also do negative indexing means if you have to find the last character in your string you can get that by -1 index as follow

```
# Find last character through negative indexing
a = 'Nomi'
print(a[-1])
```

```
# find second last character
a = 'Nomi'
print(a[-2])
```

Lists

Let assume that you want to store lots of values in one variable you can do that through list

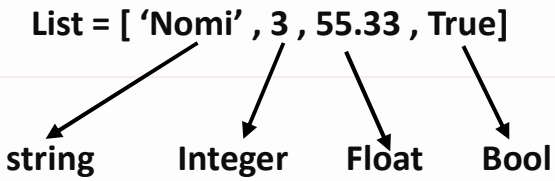
==> List is a container which can store any type of data.

==> Each value in a list can be separate with a comma.

==> List is mutable means you can change the values in a list.

==> List start and close with large brackets.

Example of List:



Practical Examples:

```
# list with different values
List_1 = ['Nomi', 33,True, 55.55]
print(List_1)
```

creating an Empty list

If you have to create a list for some purpose you can do as:

```
# Empty List
empty_list = []
```

List Indexing & Slicing

If you have to find elements through index or slicing in your list you can do that as:

```
# find value at index[0]
my_list = [1,2,3,4,5,6]
print(my_list[0])
```

```
# get values from 2 to 6 index
my_list = [1,2,3,4,5,6]
print(my_list[2:6])
```

```
# get values from 2 to 6 index
my_list = [1,2,3,4,5,6]
print(my_list[2:6])
```

```
# get last element through negative indexing
my_list = [1,2,3,4,5,6]
print(my_list[-1])
```

```
# get all elements in a list
my_list = [1,2,3,4,5,6]
print(my_list[:])
```

List Methods

Following are some methods used for a list

1.append:

This method will add elements into your existing list

Example:

```
fruit_list = ['banana', 'mangoes', 'apples']
# Add Orange into the above list
fruit_list.append('oranges')
print(fruit_list)
```

2.insert:

if you have to insert a value in a specific index you can do that by insert method by write index number followed by a value

Example:

```
fruit_list = ['banana', 'mangoes', 'apples']
# insert 'Orange' at index 0
fruit_list.insert(0, 'orange')
print(fruit_list)
```

3.Sort

If you have to sort your list in ascending or descending order you can do that by a sort method as follow:

```
# sort the following list in ascending order
list1 = [3,2,4,5,8,6,1,7]
list1.sort()
print(l1)
```

```
# sort in descending order
list1 = [3,2,4,5,8,6,1,7]
list1.sort(reverse = True)
print(list1)
```

4.Reverse

This method will reverse your entire list as we do above for sorting lets see example again

```
# Reverse the following list
list1 = [3,2,4,5,8,6,1,7]
list1.reverse()
print(list1)
```

5.Copy()

This method will make a copy of your list

Example:

```
list1 = [3,2,4,5,8,6,1,7]
# make a copy of above list
list2 = l1.copy()
print(list2)
```

6.Remove¶

This method will remove any element from your list

Example:

```
# Remove 3 from the following list
list1 = [3,2,4,5,8,6,1,7]
list1.remove(3)
print(list1)
```

Update a value in a list

If you have to replace any value with new value in a list you can do that by write variable name specify the specific index which should be replaced and finally asign the new value as follow:

```
# put 100 in place of 3
list1 = [3,2,4,5,8,6,1,7]
list1[0] = 1000
print(list1)
```

Tuples

Tuples are the same as lists in python which can store different types of data but the only difference is tuples are immutable you cannot change the elements of tuple.

Tuple start and ends with parenthesis

Examples:

```
# Create a Tuple
my_tuple = (1,'nomi', True)
print(my_tuple)
```

methods of tuples.

count

you can count the elements in a tuple like:

```
# Example
my_tuple = (1,'nomi', True)
my_tuple.count('nomi')
```

Index

you can also get the index of an element as like:

```
# Example
my_tuple = (1,'nomi', True)
my_tuple.index(1)
```

Trying to change the elements

If you try to change the elements in a tuple its will give you an error as like:

```
# Example
my_tuple = (1,'nomi', True)
my_tuple[0] = 100    # Tuple is immutable as its shows type error
```

```
# Example 2
Value1 = True
Value2 = False
print("The comparison of Value1 and Value2 is",(Value1 and Value2))
print("The comparison of Value1 or Value2 is", (Value1 or Value2))
print("Value2 is",(not Value2))
```

Special Operators

Python also provide some special operators like:

1. Identity operators

Two types of identity operators

➔**is** ==> if both values are identical to each other it will return True if not then False

➔ **is not** ==> If both values are not identical to each other it will return True otherwise False

Following are some Examples:

```
# Example
a = 23
b = 45
c = 'nomi'
d = 'sonu'
print(a is b)
print(a is not b)
print(c is d)
print(c is not d)
```

2. Membership operators

Two Types

1. in

2. not in

in check the value in a sequence if that is found it's show's True if not then False

not in check value in a sequence if that is not found it's show's True otherwise False

```
# Example
names = 'nomi', 'rizwan', 'marwan'
print('nomi' in names)
print('nomi' not in names)
```

Dictionary

Dictionary in python is the collection of key value pairs while it is used for large amount of data. ==> It is unordered
==> It is mutable means elements can be changeable
==> dictionary Cannot Contain duplicate Values ==> Dictionary is widely used for retrieving the data from dictionary through keys of the values in the dictionary.
==> dictionary store keys and values between braces {}.
==> You can store any type of data in a dictionary.
==> keys and values must be separate through a colon while each line should have a comma at the end.
Let's see some of the examples of dictionary

Create a dictionary with key value pairs

```
# Example
my_dict = {
    'name' : 'Nomi Khan',
    'Age'   : 24,
    'Degree' : 'Computer Science'
}
```

In the above Example 'name' , 'Age', and 'Degree' are the keys while 'Nomi khan' , 24 , 'Computer Science' are the values for that keys

Dictionary Methods

1. Get Key Value Pairs

If you have to get both keys and values for that keys then you can do that items() function as follow:

```
my_dict # get key and values
 = {
    'name' : 'Nomi Khan',
    'Age'   : 24,
    'Degree' : 'Computer Science'
}
print(my_dict.items())
```

2.Get Only Keys

If you have to find all the keys in your dictionary you can do that by keys() function as follow.

```
# get only values
my_dict = {
    'name' : 'Nomi Khan',
    'Age'   : 24,
    'Degree' : 'Computer Science'
}
print(my_dict.keys())
```

3.Add another key value

If you have to add new key values to your dictionary you can do that by update() function as follow:

```
# Update your dictionary
my_dict = {
    'name' : 'Nomi Khan',
    'Age'   : 24,
    'Degree' : 'Computer Science'
}
my_dict.update({'Youtube Channel': 'Nomi Intelligence'})
print(my_dict)
```

4. Get any value through Keys

If you have to get any value through keys availaible in your dictionary you can do that by get() function as follow.

```
# get name in the following dict
my_dict = {
    'name' : 'Nomi Khan',
    'Age'   : 24,
    'Degree' : 'Computer Science'
}
print(my_dict.get('name'))
```

Conditional Statements

Conditional statements are such statements through which we make a decision for example
if tomorow is rain i will not go to school, in this case rain is condition similarly in python programming following are
conditional Statements.

1. If¶
2. elif
3. Else

Let's use these with examples

Syntax:

if (condition): ⟶ **if this condition become True**

print(' yes ')

else: ⟶ **if the above condition becomes False**

print('above is false')

Practical Example if , else

check the condition between two variables

```
# check whether 21 is greater than 54 or not
a = 21
b = 54
if (a>b):
   print('yes a is greater')
else:
   print('no b is greater')
```

Output:

no b is greater

as you see in the above case 'a' is smaller the condition become false so it's gone to the else condition and print that 'b' is greater

Example elif

If you have to check multiple conditions you can do that by using elif in between if and else condition here's how:

```
# check multiple conditions
a = 21
b = 54
if (a>b):
    print('yes a is greater')
elif (a==b):
    print('yes a is equal to b')
elif (a<b):
    print('a is less than b')
elif (a>b):
    print('a is greater than b')
else:
    print('Failed')
```

The above program will run only the True condition

Use of is & in

```
# check an element in the list
a = [3,4,5,6,8]
if (5 in a):
    print('Yes')
else:
    print('No')
```

Example

Q: Write a program which take age as input from the user and apply the condition if the age is greater than 10 they can follow the course otherwise no.

```
# Solution
age = int(input('enter your age \n'))
if (age>=10):
    print('Yes you can join the course')
else:
    print('Sorry your still a kid')
```

Note:

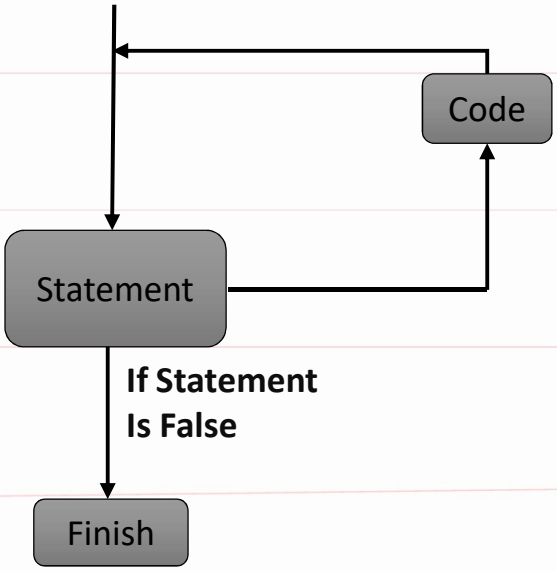
else statement is executed if all the statements become false

Loops

Loops

Loops in python can be define as the execution of a statement or multiple statements untill the given condition becomes true.

Following diagram Explain the concept of loops



Python programming provide three types of loops:

1. While loop
2. For loop
3. Nested loop

While loop¶

While loop is used to execute the statement untill it's become True. for example if you have to print integer numbers from 0 to 20 you can do it by while loop as:

```
# use while loop to print 50 numbers
i = 0  # here loop start
while (i<=20):  # Condition
    print(i)    # print numbers
    i += 1      # means that each number increase by one
```

```
# while loop i-e:2
#Print numbers upto 1000 spaced by 100
i = 100
while(i<=1000):
    print(i)
    i += 100
```

```
# Example 3 while loop
# print your name 5 times
i = 0
while(i<=5):
    print("Nomi Intelligence")
    i += 1
```

Break Statement:

if you have stop the loop even the condition becomes true you can use the break statement as follow:

```
# print numbers upto 10 but break at 5
i = 0
while (i<=10):
    print(i)
    if (i == 5):
        break
    i += 1
```

Continue Statement

If you have to stop at certein point in your code skip some values and then continue again you can use continue statement

```
# Example
# print numbers upto 10 but skip the value number 5
i = 0
while (i<=10):
        i += 1
        if (i == 5):
            continue
        print(i)
```

2.For Loop:

In python for loop is used to iterate over a value in a sequence

==> This is one of the common used loop in python

==> We can use for loop in a list, dict, tuple, set etc

for example:

```
id_no = ()
for value in id_no:
print(id_no)
```

OutPut:

12345

```
# Example for loop
# search in a list for values
my_list = [23,'nomi', 55,66,'khan']
for items in my_list:
    print(items)
```

Used Break Statement In for loop

```
# break if the biology name comes
subjects = ['chemistry', 'computer' ,
'biology' , 'maths']
for sub in subjects:
    print(sub)
    if (sub == 'biology'):
        break
```

Used Continue Statement In for loop

```
# skip 'computer' and continue with others
subjects = ['chemistry', 'computer' , 'biology' ,
'maths']
for sub in subjects:
    if (sub == 'computer'):
        continue
    print(sub)
```

Range Function in for loop

Range function is used to produced sequence of numbers

==> range function by default start from 0 while increment by 1 and ends with a specified numbers

==> you can specify the start, increment and end numbers

```
# Example
# print numbers start from
0 upto 100 increment by 10
for i in range(0,100,10):
    print(i)
```

```
# Example 2
for x in range(4):
  print(x)
```

Function's

Function in python is a block of code which is only run when required
==> When program gets better its difficult for the programmers to handle the code thus function can be used to do a specific
task for us and minimize our code.
==> A function can be resued at any number of times in a specific program
==> In python a function can be crated by write the def keyword followed by parenthesis

```
# Example
def function():
    print('Nomi Intelligence')
```

This function can be used to print a 'Nomi Intelligence' at any part of the program

Call A Function

You can call a function by just writing function name you have defined as like

```
# Call a function
def func():
    print('Hello_World')
func() # Function calling
```

Types Of Function

There are two types of function in python

1. BuiltIn Functions

Such Types of functions which are already defined in python
for example:
==> replace() function which is used to replace things in python
==> sort() function which is used for sorting
==> len() which is used to find the length

2. User Defined Functions

Such type of function which can be defined by the programmer or user with his/her own choice
user defined Function can be anything
for Example:
def My_Name():
following are the examples of built in and user defined functions

```
# Example Built In Function
a = [3,4,5,6,7,8,5]
print(len(a)) # Built In function
```

```
# Example of User Defined
Function
def my_name():
    print('Nomi_Khan')
my_name()
```

Arguments Of Function:

If you have to pass some information to a function that is called arguments
==> you can pass as many arguments you want
==> Arguments are such values through which a function can work with
==> A function must be called with the correct numbers of arguments if you have put wrong then it's goanna give an error
==> A function can also return a value
For Example:

```
# Write a function and pass a parameter name
def welcome(name):
    return 'Welcome to Nomi Intelligence ' +
name
welcome('jhon')
```

Make a function to do Addition for us

The above function will do addition for the two parameters we have given in the parenthesis if you give more than two values it's will give you an error however you can increase the parameter amount as follow:

```
# Make a function to add four numbers and find average
def average(a,b,c,d):
    avg = a+b+c+d/4
    return avg
average(33,22,12,23)
```