WELCOME TO LYRICLIOUS

Jamboard1
Jamboard2

History of Python -

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

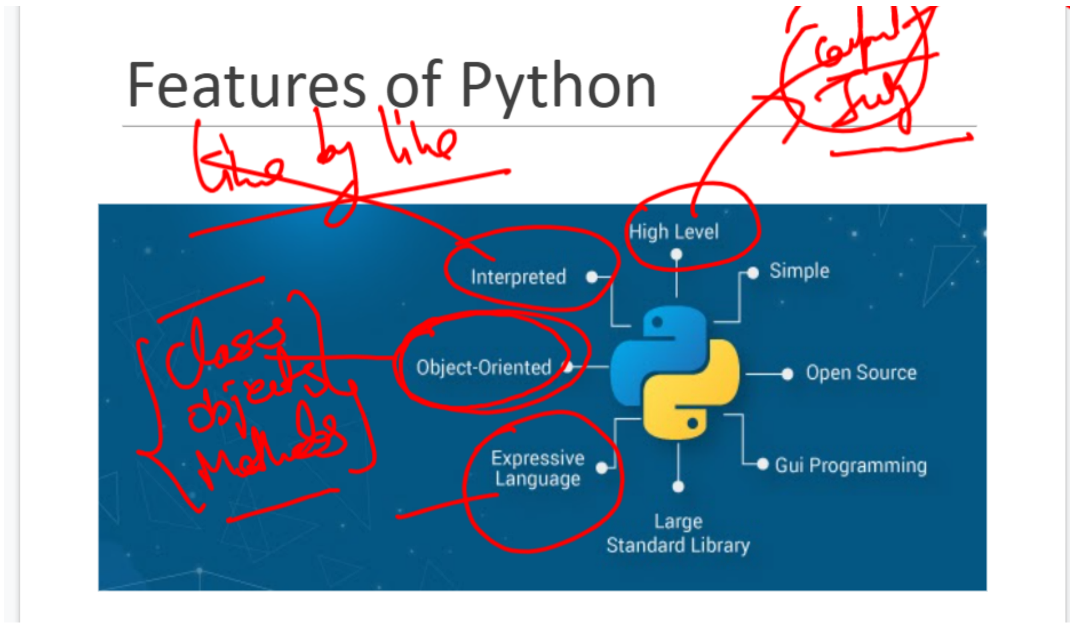
Version of Python - python 2 and Python 3

Interpreter - convert high level language into low level language line by line.

Compiler - convert high level language into low level language at once.

Features of Language -

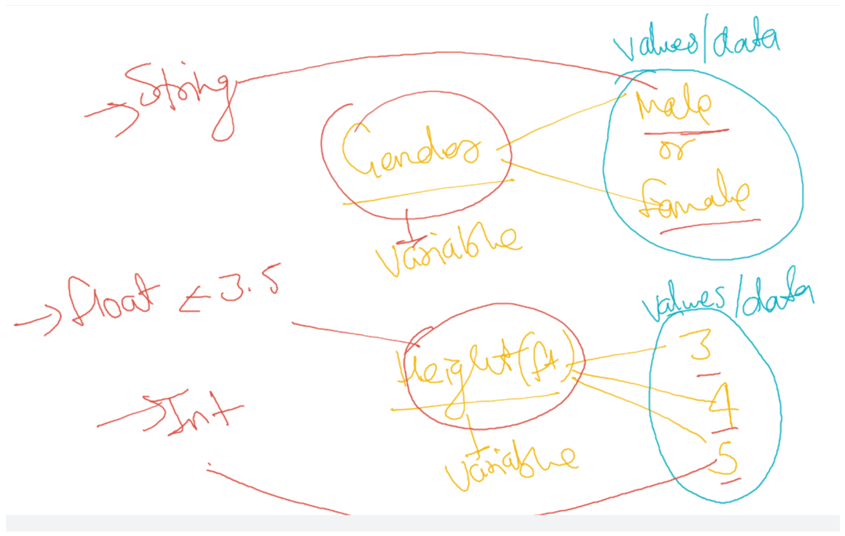
1. Python is object-oriented Structure supports such concepts as polymorphism, operation overloading and multiple inheritance.
2. Indentation is one of the greatest features in python.
3. It's free (open source) Downloading python and installing python is free and easy
4. It's easy to use and learn No intermediate compile.



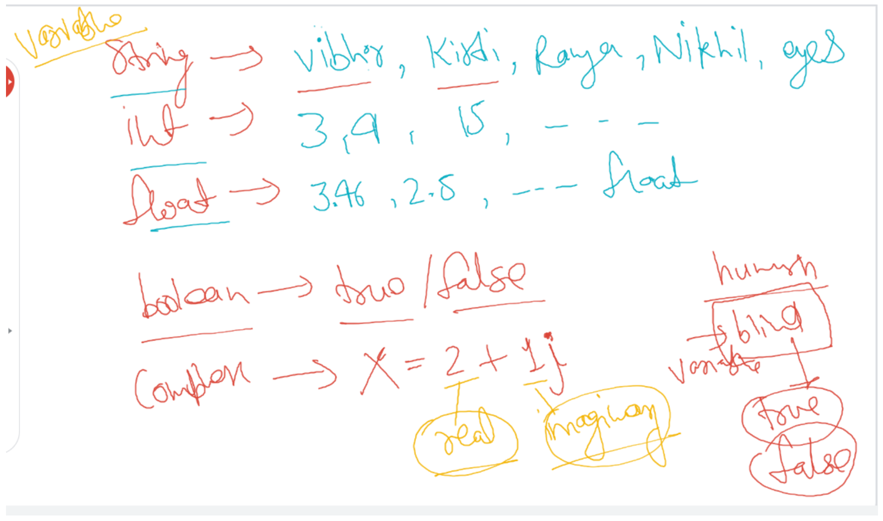
Variables -

Used as container for storing data/values.

Example -
X = 5
Y = “Happy”
print(x)
print(y)



Datatype -



Int - used to store integer values. Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Example -

```
X = 5
print(type(x))
print(x)
```

Float - Float, or "floating point number" is a number, positive or negative, containing one or more decimals. Float can also be scientific numbers with an "e" to indicate the power of 10.

Example-

```
Y = 2.5
print(type(y))
print(y)

z= 34e3
print(type(z))
print(z)
```

Complex - this type has 2 parts one is real no and other is imaginary no.

Example -

```
X = 2+1j
print(type(x))
print(x)
```

Boolean- this datatype can store only 2 values either true or false.

```
Y = True   // t should be capital.
print(type(y))
print(y)
```

String - Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes.

H , hello, hello interns , hello interns welcome to python training

Example1-

```
X = “hello interns”
print(type(x))
print(x)
```

Example2-

```
Y = “”” this is me
and my students  “””
print(type(y))
print(y)
```

0	1	2	3	4	5	6	7	8	9
L	y	r	i	c	l	i	o	u	s

Looping in string-

```
For x in “Lyriclious”:
print(x)
```

Length of String -

```
A = “Lyriclious”
print(len(x))
```

Searching in string-

```
A = “Lyriclious the best tech institute.”
print(“best” in a)
```

Or

```
A = “Lyriclious the best tech institute.”
If “best” in A:
print(“yes ‘BEST’ is present in given string”)
Else:
print(“Not present in given string”)
```

Or

```
A = “Lyriclious the best tech institute.”
If “Tasty” in a:
print(“yes ‘BEST’ is present in given string”)
Else:
print(“Not present in given string”)
```

Indexing-

X = “Hello interns”

0	1	2	3	4	5	6	7	8	9	10	11	12
H	e	l	l	o		i	n	t	e	r	n	s
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

String Indexing

Slicing-

X = “Hello interns”

1) from given start index to given end index
   print(x[2:5])

2) from start index to given index
   print(x[:5])

3) from given start index to end index
   print(x[2:])

4) Using negative index
   print(x[-5:-2])

String functions -

X = “Hello interns”

Uppercase() - convert the string in uppercase.
print(x.upper())

Lowercase() - convert the string to lowercase.
print(x.lower())

Replace() - replace the string with another string.
print(x.replace(“Hello”, “yoo”))

Split() - splits the string between the specified separator.
print(x.split(“,“))

Strip() - removes the whitespace(space) from start and end of the string.
print(x.strip())

String Concatenation-

Combine two or more strings is called as string concatenation, with + operator.

Example1-
x= “hello”
Y = “interns”
print(x+y)

Example2-
 print(x+ ” “ +y)

Example3-
Batch = 101
print(x + “ “ + y +” “ + batch)
// throws error so we cannot do it like this.

format()- this method allows us to insert numbers in string.

Example1-
Batch = 101
Text = “hello interns {}”
print(text.format(batch))

Example2-
x = 2021
y = 101
text = “Hello interns welcome to {} batch of python training {} .”
print(text.format(y,x))

Escape Character-

\’	Single quote

\\	Backslash
\n	New line
\t	tab
\b	Backspace

Example-

1) print(“We all are \”Lyriclious\” family. ”)
2) print("We all are Lyriclious\n family.")

H/w-

1) Make a list of all string functions.
2) Write a program to find the length of the string "your name" with using len function.
3) Write a program to check if the word 'name' is present in the "about yourself".

Day3-

Type Casting / type conversion - you can convert variable from one datatype to another .

Example1-

```
X = float(1)
Y = int(2.4)
Z = str(3)
print(x)
print(y)
print(z)
```

Example2 -

```
x = 1          # integer type
Y = 2.8        # float
Z = 2+1j           #complex

a= float(x)
b= int(y)
```

```
c = float(z)
print(a)
print(b)
print(c)
print(type(a))
print(type(b))
print(type(c))
```

Operators -

Arithmetic operators -

+	addition	print(3+5)
-	substraction	print(8-3)
*	multiplication	print(2*3)
/	division	print(10/2)
%	modulus	print(10%3)
** or ^	Exponent	print(3**2)
//	Floor division => rounds the result down to the nearest whole number	print(14//5)

Assignment Operators -

=	x=2	x =2
+=	X +=4	x = x+4
-=	X -=4	x= x-4
*=	X *=4	X = x*4
/=	X /=4	x = x/4
%=	X %=4	X = x%4
//=	X //=4	X = x //4

**=	X **=4	X = x**4

Comparison operators -

==	equal	2 == 3 #false
!=	Not equal	2 != 3 #true
>	Greater than	2 >3 #false
<	Less than	2<3 #true
>=	Greater than or equal to	2 >= 3 #false
<=	Less than or equal to	2 <= 3 #true

Logical Operators -

and	Returns true if both the conditions are true	2 < 3 and 3 < 10 #true
or	Returns true if any one or both the conditions are true	3 > 2 or 2 > 3 # true

not	Reverse the result	not(2 >5) #true

<u>List -</u> list are used to store multiple items in a single variables.
List are ordered , changeable and allow duplicates.

<u>Example 1-</u>
L = [“a”,”b”,”c”]
print(L)

<u>Example 2-</u>
List = [“abc”,21,True, 4.5]
print(List)

<u>Length of list</u> - print(len(L))

print(type(list))
<u>Access the item -</u>

print(list[1])
print(list[-1])
print(list[2:5])

<u>Insert items at particular place -</u>
List = [“apple”, “banana”, “cherry”]
list.insert(2,”watermelon”)
print(list)

<u>Append item at particular place -</u>
list.append(“orange”)
print(list)

<u>Remove item -</u>
list.remove(“banana”)
print(list)

<u>Remove specified item -</u>

list.pop(1)
print(list)

Or
Del list[0]
print(list)

<u>Loop through list-</u>
List = [“apple”,”banana”,”cherry”]
For x in list:
print(x)

Tuple- Example1 -

Tuples are used to store multiple items in a single variable. Tuples is ordered but unchangeable.it allows duplicates.

List = [ ]

Tuple = ( )
Tuple = (“apple”,”banana”)
print(tuple)

<u>Length of tuple -</u>
print(len(tuple))
print(type(tuple))

<u>Accessing items-</u>
print(tuple[1])
print(tuple[-1])

<u>Append tuple- </u>once a tuple is created you cannot change it directly.

<u>Example -</u>
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)

<u>Insert / add items</u> - once a tuple is created you cannot add items to it.

<u>Example -</u>
x = ("apple", "banana", "cherry")

```
y = list(x)
y.append("kiwi")
x = tuple(y)
```

<u>Remove/ delete items-</u> once a tuple is created you cannot delete the item directly.

<u>Example -</u>

```
x = ("apple", "banana", "cherry")
y = list(x)
y.remove("kiwi")
x = tuple(y)
```

<u>Looping through items of tuple-</u>

<u>Example1-</u>

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

<u>Set operations-</u>

Let set A and set B

```
union()
intersection_update()
intersection()
symmetric_difference_update()
symmetric_difference()
```

<u>Dictionary-</u>

They are used to store data values in key,value pair.
They are ordered,changeable and does not allow duplicates.

<u>Example-</u>

```
Dict = {
“Brand”:”ford”,
“Model”:”mustang”,
“year”:1964
}
print(dict)

print(len(dict))
print(type(dict))
```

<u>Access values and keys-</u>

```
print(dict.values())
print(dict.keys())
```

<u>Append dictionary -</u>
<u>Example-</u>

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.values()

print(x)         #before the change

car["year"] = 2020

print(x)         #after the change
```

<u>Loop through dictionary-</u>
<u>Example-</u>

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

<u>Update items-</u>
<u>Example -</u>

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
```

Remove dictionary-

Example-

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

Python Conditions and If Statements -

Conditions -

Equals : a == b
Not Equals : a != b
Less than: a <b
Less than or equal to : a <= b
Greater than : a > b
Greater than or equal to : a >= b

If statement syntax-

```
If condition:
        Statement 1
```

If-else statement syntax-

```
If condition :
     Statement 1
     Statement 2
Else:
     Statement 1
```

Else cond catches anything which is not caught by preceding cond

```
If condition1 :
     Statement 1
     Statement 2
Elif condition2 :
     Statement 1
     Statement 2
```

```
Elif condition3 :
    Statement 1
    Statement 2
Else :
    Statement
```

<u>While Loops-</u>

We can execute a set of statements until a condition is true.

```
While #condition:
```

<u>For loops-</u>

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

Range()-
The range function returns seq of number starting from 0 by default and inc by 1 & stop before specified number

Functions

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or can not return one or more values.

```
def functionName(parameter1, parameter2):
        Set of instructions
```

Calling functions:

```
functionName(argument1, argument2)
```

<u>Types of Functions -</u>

There are two types of function -
1) Built-in function
2) User defined function

What is recursion?

Recursion is the process of defining something in terms of itself.

A physical world example would be to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.

We know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.

5! = 5 X 4! = 5 X (4 X 3!) = 5X4X(3X2!)= 5X4X3X2X1!=5X4X3X2X1

N! = n X(n-1)!

Ex 1- Write a Python program to calculate the factorial value

```
def recursive_factorial(n):
   if n == 1:
       return n
   else:
       return n * recursive_factorial(n-1)

num = 6
if num < 0:
   print("Invalid input ! Please enter a positive number.")
elif num == 0:
   print("Factorial of number 0 is 1")
else:
   print("Factorial of number", num, "=", recursive_factorial(num))
```

Ex 4 - Write a program to compute the square of a number using recursive function.

```
def findSquare(targetNumber) :

  if targetNumber == 0 :
    return 0
  else :
    return findSquare(targetNumber - 1) + (2 * targetNumber) - 1

targetNumber = int(input(“Enter the number”))
print(findSquare(targetNumber))
```

Python Sorting Method -

The sort() method sorts the list ascending by default.

Syntax- list.sort(reverse=True|False)

Example1 -
```
cars = ['Ford', 'BMW', 'Volvo']

cars.sort()
```

This sort the array in ascending order.

Example2 -

```
cars = ['Ford', 'BMW', 'Volvo']

cars.sort(reverse=True)
```

This sort the array in descending order.

Types of sorting Techniques-

Insertion Sort
selection Sort
Merge sort
Shell Sort
Bubble sort
Quick sort
Heap sort
Counting Sort
Radix Sort
Bucket Sort

Gnome Sort
Comb Sort
Shell Sort

Python Modules -

A file containing a set of functions you want to include in your application.

Import math

Create a Module -

Few Popular Modules are -

1) Python Datetime - A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

   Example -

   ```
   import datetime

   x = datetime.datetime.now()
   print(x)
   print(x.year)
   ```

   Output - it will be in format of 2021-06-02 13:51:59.264423
   The date contains year, month, day, hour, minute, second, and microsecond.

2) Python Math -   Python has a set of built-in math functions, including an extensive math module, that allows you to perform mathematical tasks on numbers.

   a) min() and max() - The min() and max() functions can be used to find the lowest or highest value .
      Example -
      ```
      x = min(5, 10, 25)
      y = max(5, 10, 25)
      ```

```
print(x)
print(y)
```

b) pow()- The pow(*x*, *y*) function returns the value of x to the power of y (xy).
Example-

```
x = pow(4, 3)

print(x)
```

Tkinter-

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method.

To create a tkinter app:

1) Importing the module - tkinter
2) Create the main window
3) Add any number of widgets to main window
4) Apply the event Trigger on the widgets

- Import tkinter
- root=tkinter.Tk()   ///where root is the name of main window
- root.mainloop()

Label - It refers to the display box where you can put any text or image which can be updated any time as per the code.

Syntax -

label=Label(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget.

- bg- to set the normal background color
- command- to call the function
- font- to set the font on the button label
- image- to set the image on the button
- Width- to set the width of the button
- height- to set the height of the button

Example -

```
From tkinter import *

root= Tk()

label=Label(root, text='Lyriclious!')

label.pack()    //grid , pack and place, these are layout management

root.mainloop()
```

Strftime - convert the date and time into string representation.