1. **Variables: -**

Variable is like a whore, jo mil jaye voh uski, which means firstly a = “Nayan”, so a = “Nayan”, later, when assigned a = “Lokesh”, so changed the value to Lokesh, later when assigned a = ‘Thorat”, so will change it to “Thorat”.

1. **Data types: -**

Different types hai jaise:

1. String – str : “Hello World!!”
2. Integer – int : 14
3. Boolean – bool : True
4. Float – float : 3.14
5. Complex – complex : 3-4i
6. **Comments: -**

(#) is used for comments

1. **Arithmetic operators: -**

+, -, \*, /

1. **Relational operators: -**

<, >, == , !=

|  |  |
| --- | --- |
| A < B | B is greater than A |
| A > B | A is greater than B |
| A == B | A is equal to B |
| A != B | A is not equal to B |

1. **Logical operators: -**

& - and operator, | - or operator

Primarily used for Boolean but can also be used in various other data types.

**- Boolean:**

A = true

B = false

A & A = True, rest all are false

B | B = False, rest all are true

**-Other data types:**

X = 40

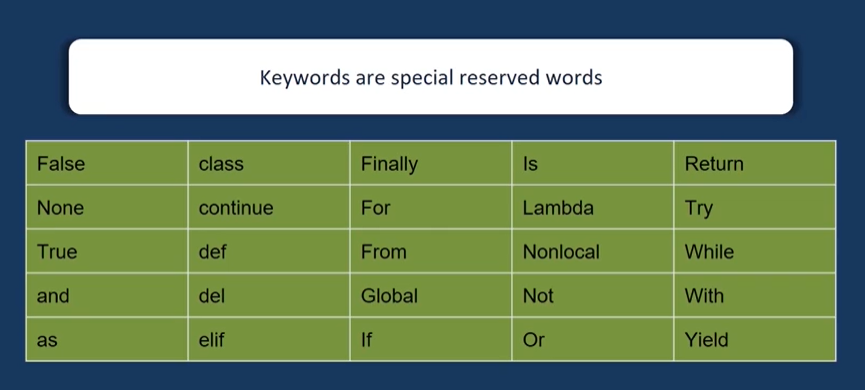
Y = 50

X > 10 & Y < 100 = True, rest all will be false

* Jab bhi aap koi input loge user se toh voh as a string save hongi, toh agr aapko koi specific operations use karne hai uspe toh aapko uske data type ko dhyan me rkhna honga.
* print(“Sum is”, sum) ---- Yeh comma ( , ) dhyan me rkhna.

* To use date and time concept in python, we need to import the datetime module, there are two ways for the same, firstly**, import datetime** in that in every command hume datetime datetime do baar likhne padenge.
* For example, for current date and time: datetime.datetime.now()
* Secondly, use **from datetime import datetime** in that we don’t add datetime datetime again and again, sirf ek hi baar likhne se ho jayega.
* For example, for current date and time: **datetime.now()**
* Parentheses – (), dekho ab iska explanation toh likha hai ki these are used to invoke a method or function, but Mujhe iska simple explanation yeh lag raha ki combination of components like date – YYYY, MM, DD and time – HH, MM, SS.
* datetime.now() is used to get current date and time combinely.
* If we only want date in that so we can do **datetime.now().date()** or we only want time in that so we can do **datetime.now().time()**
* If we only single component like hour or month, so here, we use **datetime.now().hour** to get the info about hour from overall time, we also use **datetime.now().month** to get the info about month from overall date.
* To use the value of pi or I guess other mathematical operations, we have to write “import math” at the top of the code, to import all mathematical values and constants, and to write pi we’ve to write it like “math.pi”. if we want to use only pi from math, then we can also use **from math import pi**, we mean we want only value of pi from math module, so here we don’t have to write math.pi instead we can write **pi** directly.
* Abhi prime number ke algo ka logic batau toh, ek for loop me range dedo i ki, from 2 to (square root of number itself + 1), aur number ko iss i se modulus function me daalo, remainder agr zero aaya, toh number is not a prime number, and vice versa.

1. **Python Keywords: -**



Keywords cannot store any value like variables.

1. **Python Identifiers: -**

Identifiers are the names used for variables, functions or objects.

Some rules of identifiers:

1. No special character expect underscore ( \_ ).
2. Identifiers are case sensitive which means Student & student are two different names for it.
3. First letter cannot be a digit.
4. **Python Literals: -**

Literals means the constant in Python.

We can also say that the value stored in variable are also known as Literals.

* ( \n ) is used to end the line and start from the new one.
* ( \t ) is used to give space as of 1 TAB ( Tab button wala ).
* import sys is used to get info about python interpreter.
* To get Python version (sys.version)
* To get Python version info (sys.version\_info)

1. **Introduction to Python Strings: -**

Strings are sequence of characters enclosed within quotes which can be single(‘ – ‘) or double(“ – “) quotes and triple quotes (‘’’ – ‘’’, “”” – “””).

Single and double are treated equivalent in python, they depend on personal preference.

Triple are used for multiline strings.

For example, ‘Hello world!!!’ or “Hello world!!!”, they both are equivalent.

‘’’ Hello world,

Nayan this side.

Pleased to meet you. ‘’’

Some string operations: -

1. Suppose there is a string -> str1 = ‘Hello world, Nayan this side!!’

So, str1[0] means the character present on index – 0 in the string – str1.

* Indexing in the string starts from 0 to n.
* Index – [ -1 ] means from end, the 1st character.
* Agr value minus me hai, toh end string se indexing 0 se start nahi hoti, end se 1 se hi string ki indexing start hoti.

1. We can also use indexing to indicate a part of the string, using str[ 2:10 ]

* Note, in str[2:10], index 2 se string se start hongi, but index 9 pe end hongi, naa ki 10 pe. REMEMBER THAT!!

In str[2:10], **2** represents the lower limit, **(10-1)** represents the upper limit.

1. Length of the string:

* We have to use, **len(str1)**

1. To rewrite the whole string in lowercase:

* We have to use, **str1.lower()**

1. To rewrite the whole string in uppercase:

* We have to use, **str1.upper()**

1. To replace some specific characters by any different character:

* We have to use, **str1.replace(‘old character’, ‘new character’)**

1. To count a character or a word in a string

* We have to use, **str1.count(“The character”)**
* **Note this counting method, is case sensitive.**

1. To know the index value of a specific character or word:

* We have to use, **str1.find(“Character/ Word”)**
* If the character or the word does not present in the string, it concludes with -1 as a value.

1. To concatenate two strings:

* We have to use, **New string name = str1 + str2**

1. To reverse a string:

* We have to use, **New string name = str1[::-1]**

1. **Basic Data Structures in Python: -**

There are four basic data structures in Python:

1. Tuple
2. List
3. Dictionary
4. Set

**Tuple: -**

Tuple is an ordered collection of elements enclosed within (). It can store heterogeneous elements which means we can store elements of different data type in a tuple.

* Yaha par bhot saare elements aapko ek saath store karna padega, isliye yaha data structures like tuple use hote hai.

Tuples are immutable which means they cannot be changed once they are created.

Ex: - tup1 = (3.14, 101, ‘Nayan’, False, (3-4i), ‘b’)

1. Extracting individual elements from tuple:

* We have to use **tup1[n]**
* The indexing of tuple starts with 0.
* It can also extract a row of tuple by **tup1[2:5]**, this is similar to string, the index 5 element will be excluded, so it was extract from index 2 to index 4.

1. Finding length of the tuple:

* We have to use **len(tup1)**

1. Concatenating (Adding) of tuples:

* We have to use **tup1 + tup2**

1. Repeating of tuples:

* We have to use **tup1\*4**

1. To find minimum and maximum values of tuple:

* For min, we have to use **min(tup1)**
* For max, we have to use **max(tup1)**

**Lists: -**

List is an ordered collection of elements enclosed within [], it is mostly same as tuple with a one major change which is, Lists are mutable which means lists can change and modify its values.

Example: - l1 = [304, ‘Nayan’, 5-4j, True]

1. Extracting individual elements:

* We have to use **l1[1]**
* For extracting a group of elements from the list we have to use **l1[2:7]**, again the same thing, element on 7th index is excluded, it results only till 6th element.

1. Changing a value of an element in the list:

* We have to use **l1[n] = 100(or whatever)**, here n is the index number.

1. Appending a new element: (Inserting)

* We have to use **l1.append(Whatever)**, by this command the new element will be added at the last.

1. Popping an element: (Deleting)

* We have to use **l1.pop()**, by this command the element at the last will be deleted.

1. Reversing elements in the list:

* We have to use **l1.reverse()**

1. Inserting an element in an specified index:

* We have to use **l1.insert(“index number”, “The value to be inserted”)**
* In that case, rest of the elements ek ek aage bad jayenge.

1. Sorting a list in alphabetical order:

* We have to use **l1.sort()**

1. Concatenating (Adding) two lists: -

* We have to use **l1 + l2**

1. Repeating of elements in a list: -

* We have to use **l1\*n**, where “n“ is the number of times.

1. To get sum of the integers in the list: -

* We have to use **sum(l1),** where l1 is the list name.

**Dictionary: -**

Dictionary is an unordered collection of key-value pairs enclosed with {}.

They are mutable in nature, which means their values can be changed.

Example: -

d1 = {‘Apple’:20, ‘Mango’:100, ‘Guava’:420}

1. Extracting only keys from dictionary:

* We have to use, **d1.keys()**

1. Extracting only values from dictionary:

* We have to use, **d1.values()**

1. To add a new element in a dictionary:

* We have to use, **d1[‘New key name’] = Value**

1. To modify a value of an existing key:

* We have to use, **d1[‘Key name’] = New value**

1. To append two different dictionaries:

* We have to use, **d1.update(d2)**

1. To pop (remove) an key:

* We have to use, **d1.pop(‘Key we want to pop out’)**

1. To retrieve the value associated with the given key:

* We have to use, **d1.get(“The key name you want the value of”, Default value)**
* The **default value** is displayed, when the given key name is not present in the dictionary. It can be anything, int/string/char/ or any other data type.

1. **If statement: -**

Syntax of if statement:

***if (condition):***

***{***

***(Blah blah blah)***

***}***

Syntax of if-else statement:

***if (condition):***

***{***

***(blah blah blah)***

***}***

***else:***

***{***

***(blah blah blah)***

***}***

Syntax of if-elif-else statement:

***if (condition):***

***{***

***(blah blah blah)***

***}***

***elif (condition):***

***{***

***(blah blah blah)***

***}***

***else:***

***{***

***(blah blah blah)***

***}***

1. **While loop:**

Syntax of while loop:

***while (condition):***

***{***

***(Executing statements)***

***}***

* Jab tak condition true hoti rahengi, tab tak while loop ke undr wale statements execute hote rahenge, jab false ki value aayi, loop exit ho jayega.

1. **Functions:**

Functions means the block of code used to perform a specific task.

Syntax to define a function:

***def function\_name():***

***(Function Body)***

***}***

To print what’s inside the function:

***function\_name()***

**Lambda function:**

This is also one of the types of function. This one is anonymous one. Usually shorter than the normal one, and is used for small tasks to be performed quickly.

Syntax:

variable\_name = lambda x: Expression

print(variable\_name (x))

* Types of Lambda function:

1. **Lambda with filter: -**

Used to filter some elements out with some expressions.

result = type\_store\_in(filter(lambda x: Expression, type\_in\_which\_the\_data\_is\_stored))

Example:

***l1 = [12,45,78,89,56,23,70,80,90,455]***

***l2 = list(filter(lambda x: x%2!=0,l1))***

***print(l2)***

***l2 = [45, 89, 23, 455]***

1. **Lambda with map: -**

Used to map all the elements with some specific expression.

result = type\_store\_in(map(lambda x: Expression, type\_in\_which\_the\_data\_is\_stored))

Example:

***l1 = (1,4,7,8,5,2,3,6,9)***

***l2 = list(map(lambda x: x\*\*2, l1))***

***print(l2)***

1. **Lambda with reduce: -**

def nearest\_power(A, B):

# Example usage:

A = 2

B = 4

result = nearest\_power(A, B)

print(result)

1. **Array:**

Array should be written in square brackets [ ].

* To take input from user

Syntax:

*arr = (input("Enter the numbers in array: "))*

*arr1 = arr.split(‘,’) ---- Used to separate elements using ( , ) if we left it empty then it will be separated by “space”*

*arr1.sort()*

* To sort the array in ascending order

*original\_array.sort()*

* To sort the array in descending order

*original\_array.sort(reverse=True)*