Python Tokens:

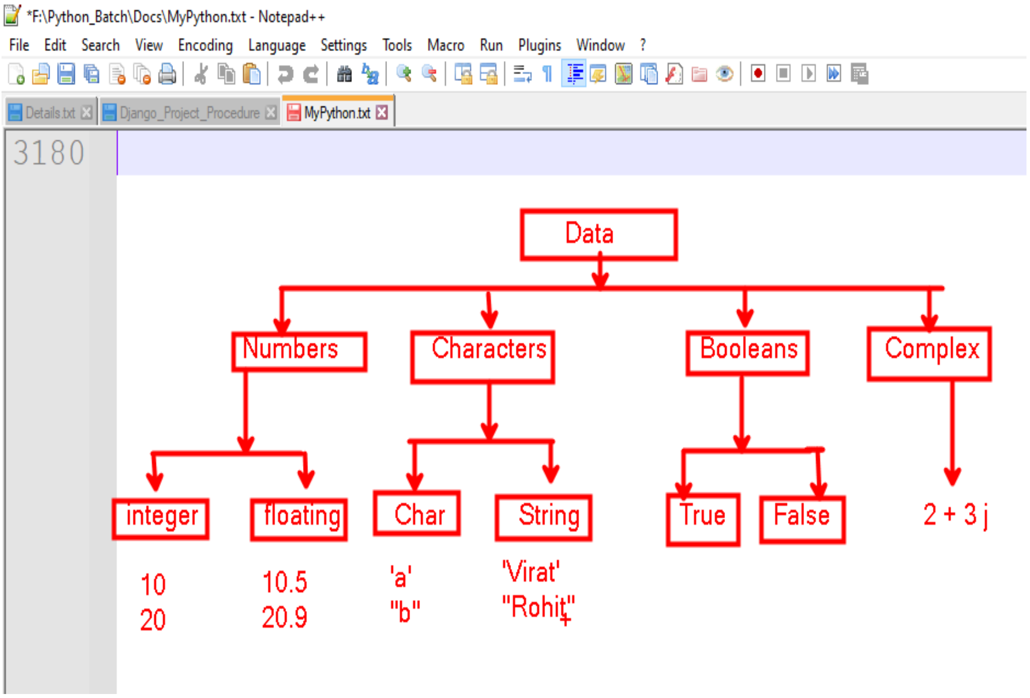
Tokens are basic elements which are used to develop Python Programs.

We have different types of tokens.

1. Literals.
2. Constants.
3. Keywords.
4. Identifiers.
5. Variables.
6. Comments
7. Indentation
8. Operators.

1. Literals.

- In python data is technically called as literals.
- Literals are classified into different types they are like below,



a) integer literals ---->> 10, 50

b) floating literals ---->> 10.30, 30.60

c) charecter literals ---->> 'a', 'B'

d) String literals ---->> 'Hello', 'Srinivas'

e) boolian literals ---->> True / False

f) complex literals ---->> 2+3j, 3+4j

2. Costants:

- ➢ The values which are fixed are called as Constants.

P.I = 3.14

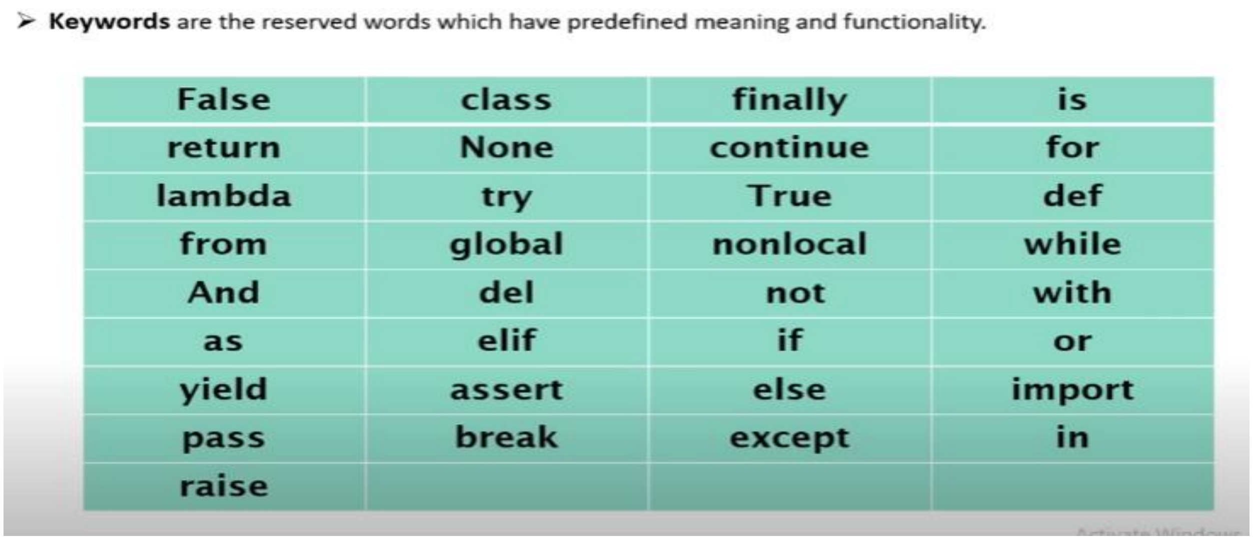
maximum_marks = 100

sun_raise = 'east'

3. Keywords:

- ➢ Keywords are also called as reserved words and pre-defined words.
- ➢ Each and every keyword has special meaning , we cannot change that meaning.
- ➢ To see all available keywords in python then open python prompt type like bellow

Diagram:



➢ **Keywords** are the reserved words which have predefined meaning and functionality.

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

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while',
'with', 'yield']
```

If we want to know totally how many keywords are avialble in python then using len() function we can find out like below,

```
>>> len(keyword.kwlist)
35
```

Note: Out of 35 keywords , only 3 keywords are starts with capital letters. They are False , True , None

4. Identifiers:

- Identifiers are user-defined words.
- Identifiers are used to identify variables, functions, methods, classes, modules and packages.

For example : a = 10, course = 'Python', cost = 70.50

Identifiers rules:

- Spaces not allowed in between Identifiers.
  For example, first name = 'Srinivas' ---->> False
- Special characters are not allowed except underscore symble
  For example, first_name ----> True
  first-name, first$name ---->> False
- Keywords are not allowed as Identifiers.
  For example, def = 10 --->> False, class if: ----->> False
- Starting character should be alphabet or underscore only but not digits.
  For example, name1 = 'Srinivas' --->> True
  _name = 'Virat' --->> True
  1name = 'Rohit' --->> False

Note: **Indentifiers can have finally alphabets or underscore or digits combination only but not others.**

5. Variables:

- Variables are nothing but reserved memory locations to store values.
- This means that when you create a variable you reserve some space in memory.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

- Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Assigning Values to Variables:

- Python variables do not need explicit declaration to reserve memory space.
- The declaration happens automatically when you assign a value to a variable.
- The equal sign (=) is used to assign values to variables.
- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

Syntax1: **variable_name = variable_value**

For Example: a = 10 , name = "Vaasu"

Syntax2**:**

variable_name = variable_value | variableReference | expressionResult | functionResult

For Example,

```
x = 10      # variable_value  assign
y = x       #  variable assign
z = x + y   # expression assign
```

Q. How to store Single value to Multiple Variables ?

a = b = c = 1

- Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.

Q. How to store multiple values into multiple variables ?

a, b, c = 1, 2, "Vaasu"

- Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "Vaasu" is assigned to the variable c.

Variables Rools:

- Variables names must be start with a letter or an underscore but not be digites ,

- The remainder of your variable name may consist of letters, numbers and underscores.
- python identifiers are case sensitive. means 'a' not equal to 'A'

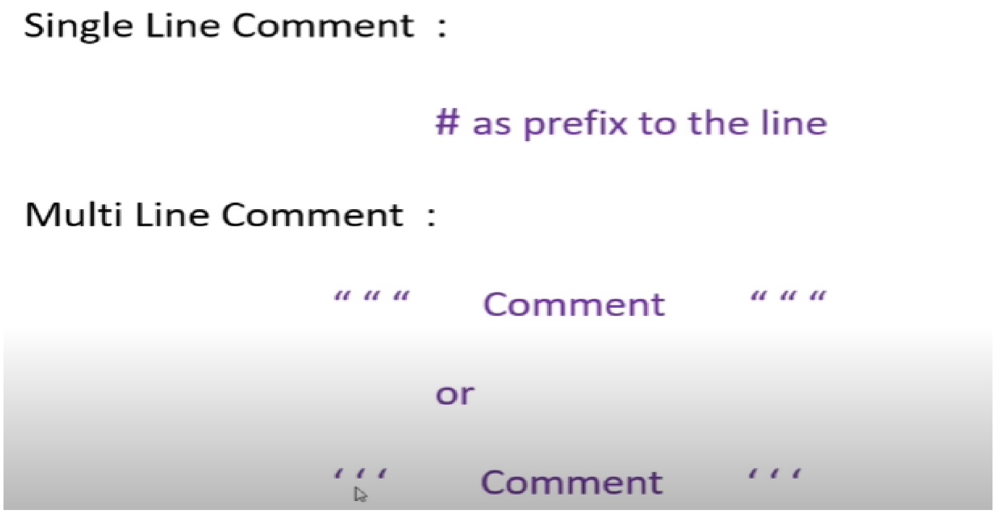
For examles :

```
>>> a = 10  ,
>>> a1 = 10  ,
>>> first_name = "Vaasu"
>>> 2a = 20     #  SyntaxError  :  invalid syntax
```

6. Python Comments:

- Python Comment is an essential part for the programmers.
- Comments are generally used to explain the code.
- We can easily understand the code if it has a proper explanation.
- A good programmer must use the comments because in the future anyone wants to modify the code as well as implement the new module; then, it can be done easily.
- In the other programming language such as C++, It provides the // for single-lined comment and /*.... */ for multiple-lined comment.
- But python provides hash (#) symble for single-lined comment and triple single or double quotes for multi-lined comment.

Diagram:



Single Line Comment :

as prefix to the line

Multi Line Comment :

“ “ “   Comment   “ “ “

or

‘ ‘ ‘   Comment   ‘ ‘ ‘

For example for single-line comment,

# Variable a holds value 5

```
# Variable b holds value 10
# Variable c holds sum of a and b
# Print the result
```

Example :

```
>>> a = 5
>>> b = 10
>>> c = a+b
>>> print("The sum is:", c)
```

- The above code is very readable even the absolute beginners can understand that what is happening in each line of the code. This is the advantage of using comments in code.

Example for multi-line comment,

```
'''
Variable a holds value 5
Variable b holds value 10
Variable c holds sum of a and b
Print the result
'''
```

Example:

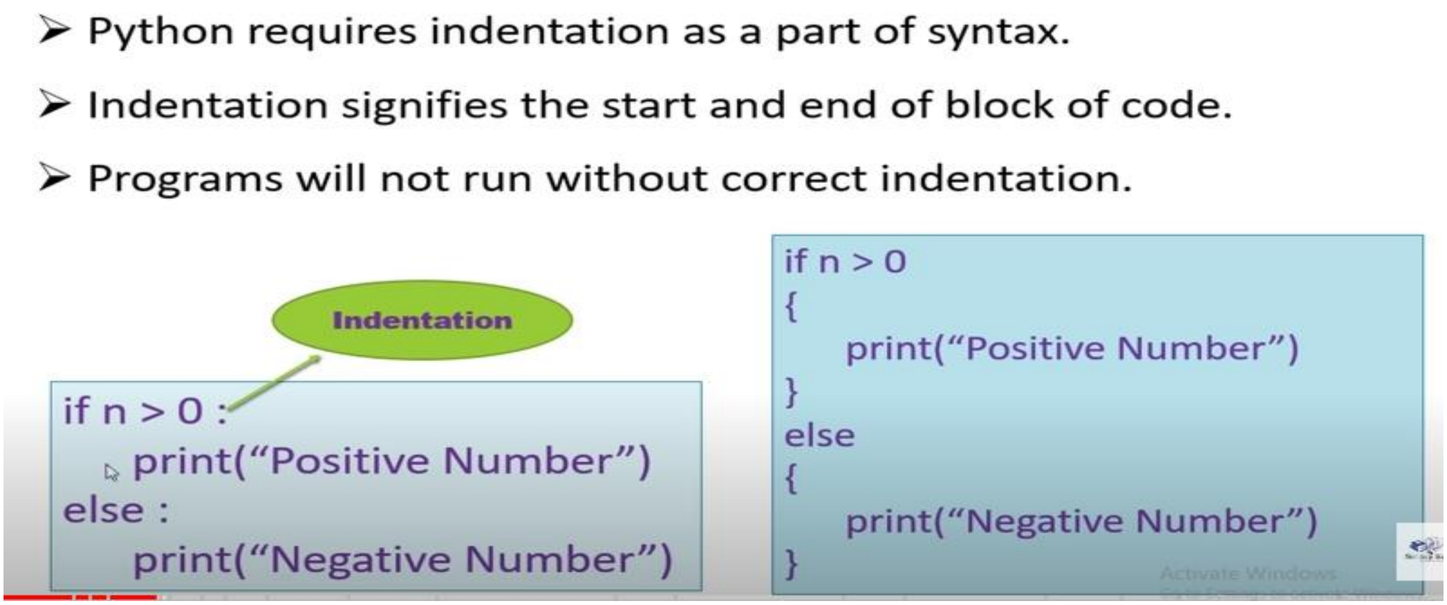
```
>>> a = 5
>>> b = 10
>>> c = a+b
>>> print("The sum is:", c)
```

7. Python Indentation :

- Python indentation uses to define the block of the code.
- The other programming languages such as C, C++, and Java use curly braces {}, whereas Python uses an indentation. Whitetspace uses as an indentation in Python.
- Indentation uses at the beginning of the code and ends with the unintended line. That same line indentation defines the block of the code (body of a function, loop, etc.)

- Generally, four whitespaces are used as the indentation. The amount of indentation depends on user, but it must be consistent throughout that block.

Diagram:



- Python requires indentation as a part of syntax.
- Indentation signifies the start and end of block of code.
- Programs will not run without correct indentation.

```
if n > 0 :
     print(“Positive Number”)
else :
     print(“Negative Number”)
```

```
if n > 0
{
     print(“Positive Number”)
}
else
{
     print(“Negative Number”)
}
```

For example,

```
 def myfunction():
     a = 10
      b = 20
      c = a + b
      print("The result is :",c)
print('out side of indentation')
```