In [1]:
```
print('hello')
```

```
hello
```

h1

h2

h3

h4

h5

h6

In [2]:
```
print('hii')
```

```
hii
```

Basic Components of Python:-

1. Comments
2. Keywords
3. Identifiers
4. Variable
5. Datatype
6. Input/Output
7. Operators

1. comments:-

In [1]:
```
# this is a single line comment
```

In [2]:
```
"""this
is
a
multi-line comment
"""
```

Out[2]:
```
'this\nis \na \nmulti-line comment\n'
```

In [3]:
```
90+89
```

Out[3]:
```
179
```

2. Keywords:- Reserved Words

In python there are 35+ keywords.

In [4]:
```
help('keywords')
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

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

In [ ]:
```
finally  # words which are green and bold are keywords
print()  # words which are green and have () are predefined functions.
var  # words which are black are variables.
'hii' # words whose color is red are valid strings.
"hello"
'''hello world'''
"""hi hello"""
```

3. Identifiers:- are the name we used give to the variables, function inorder to identify.

Rules of Naming an Identifier:- "Identifier's Rule"

In [5]:
```
# 1. We cannot use keywords are identifier's name.
except = 100
```

```
  Cell In[5], line 2
    except = 100
    ^
SyntaxError: invalid syntax
```

In [6]:
```
# 2. We cannot use digits as the first character of an identifier's name.
12var = 'apple
```

```
  Cell In[6], line 2
    12var = 'apple
     ^
SyntaxError: invalid decimal literal
```

In [8]:
```
# 3. We cannot have spaces in between identifier's name.
var one = 'apple'
```

```
  Cell In[8], line 2
    var one = 'apple'
        ^
SyntaxError: invalid syntax
```

In [10]:
```
# 4. We cannot use any special characters except an underscore.
var#1 = 'apple'
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 2
      1 # 4. We cannot use any special characters except an underscore.
----> 2 var

NameError: name 'var' is not defined
```

In [11]:
```
var_1 = 'cat'
```

In [12]:
```
# 5. python is a case-sensitive language i.e: 'a' != 'A'.
v = 'cat'
print(V)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[12], line 3
      1 # 5. python is a case-sensitive language i.e: 'a' != 'A'.
      2 v = 'cat'
----> 3 print(V)

NameError: name 'V' is not defined
```

Always start an identifier's name with alphabet which can be followed by digits or can use only '_' to separate multiple words.

In [13]:
```
var_one = 'pine'
```

4. Variables:- are like a container that helps to store values which is also used for memory allocation.

```
Variables always follows Identifier's Rule while naming.
```

In [14]:
```
var = 1000
```

In [15]:
```
print(var)
```

```
1000
```

5. Datatypes:-

In [16]:
```
# 1. int:-
var1 = 2189734979743789
type(var1)
```

Out[16]: int

In [17]:
```
# 2. float:-
var2 = 6778789.99
type(var2)
```

Out[17]: float

In [18]:
```
# 3. complex:-
var3 = 678+77j
type(var3)
```

Out[18]: complex

In [19]:
```
var4 = 88+89i
type(var4)
```

```
  Cell In[19], line 1
    var4 = 88+89i
                 ^
SyntaxError: invalid decimal literal
```

In [20]:
```
# 4. bool:-
var5 = True
type(var5)
```

Out[20]: bool

In [21]:
```
# 5. string:-
var6 = 'hello'
type(var6)
```

Out[21]: str

In [22]:
```
var7 = "A"
type(var7)
```

Out[22]: str

In [23]:
```
var8 = '''world'''
type(var8)
```

Out[23]: str

In [24]:
```
var9 = """hello
world"""
type(var9)
```

Out[24]: str

In [25]:
```
# Sequential datatype:-
# 1. list:- are sequence of elements present within [ , , ,].
var10 = [1,2,3,4,5,76]
type(var10)
```

Out[25]: list

In [26]:
```
var11 = ['apple',78,True, 78+8j]
type(var11)
```

Out[26]: list

In [27]:
```
# 2. tuple:- are sequence of elements within (, , ,).
var12 = ('apple','bat', 'van')
type(var12)
```

Out[27]: tuple

In [28]:
```
var13 = ('Van', 12, 90.99,False)
type(var13)
```

Out[28]: tuple

In [29]:
```
# 3. set :- are sequence of elements within {, , ,}.
s1 = {1,23,4,6,7}
type(s1)
```

Out[29]: set

In [30]:
```
s2 = {'world' , 67, 89.44, True, 90+34j}
type(s2)
```

Out[30]: set

In [33]:
```
# 4. dictionary:- are collections of paired element denoted with {}.
# d = {keyname : value, keyname : value}
# keynames can be made using any primary datatypes
# values can be made using any primary or sequential datatypes.
d = {1:'apple',67.99:[3,4,6,78],90+7j:True}
type(d)
```

Out[33]: dict

Input :- are used to make our programs interactive as well as to make the correctness of our program.

In [36]:
```
name = input("Enter your name: ")
```

In [37]:
```
name
```

Out[37]: 'lily'

In [38]:
```
a = input("Enter a number: ")
b = input("Enter 2nd number: ")
print(a+b)
```

```
4510
```

In [39]:
```
type(a)
```

Out[39]: str

Typecasting:- a process of converting one's default datatype to another datatype explicitly.

In [40]:
```
a = int(input("Enter a number: "))
b = int(input("Enter 2nd number: "))
print(a+b)
```

```
55
```

In [41]:
```
type(a)
```

Out[41]: int

Output :- to check the result or the outcome of our program.

In [42]:
```
print("This is my message")
```

```
This is my message
```

In [46]:
```
a = int(input("Enter a number: "))
b = int(input("Enter 2nd number: "))
print("The sum of",a,"and", b,"is", a+b)
```

```
The sum of 45 and 10 is 55
```

In [47]:
```
# string formatting:- a way to combine message along with expressions
a = int(input("Enter a number: "))
b = int(input("Enter 2nd number: "))
print(f"The sum of {a} and {b} is {a+b}")
```

```
The sum of 45 and 78 is 123
```

In [ ]: