Hello World

#we will use the print()function to print our first python program:'hello world'. #Here, we 'hello world'is a string ,a sequence of characters.In python,string are enclosed in single quotes,double quotes, or triple quotes

In [1]:

```
print('hello world')
print("hello world")
#triple quotes allows quotes to spam triple lines
print("""hello


world""")
```

```
hello world
hello world
hello


world
```

1.2 Comments

#comments are very important while writing a program .They explains the code in line for programmers to better understand program .In python we use the hash (#)symbol to writing A Comment .Python Interpreter ignores comments during execution. .# for one lines #We can have comments that extend up to multiple lines .One way is use to the hash(#)symbol at the begining of each line.Another way of doing this is use to triple quotes ,either '''or'''. .#for multiple lines

In [2]:

```
print('hello world')#this is a comment
```

```
hello world
```

In [3]:

```
'''
triple quotes
for multiple

lines


'''
print('Hello World')
```

```
Hello World
```

1.3 Packages and Modules

#Python has packages and modules .A modules is a file cotaining Python code. A python is like a directly that contains sub package is like a directory that contain sub package and modules . A library contains multiple modules ,As our application program grows larger in size with a lot of modules ,we place similar modules in one package and different modules in different packages.This makes a project(program)easy to manage and conceptually clear.Python can have sub packages and modules. #to use these packages and modules in Python; .# import .# from...import

In [4]:

```
import os#import os module
os.getcwd()# get the current  working directory
```

Out[4]:

```
'C:\\Users\\DElll'
```

In [5]:

```
from os import getcwd
getcwd()
```

Out[5]:

```
'C:\\Users\\DElll'
```

#if a package name is too complex to call ,you can give it an alias

In [6]:

```
import platform
platform.platform()
```

Out[6]:

```
'Windows-10-10.0.19044-SP0'
```

In [7]:

```
import platform as pf
pf.platform()
```

Out[7]:

```
'Windows-10-10.0.19044-SP0'
```

1.4 Variables

#variables are use for storing data values.A variables is an named location to store data in memory

In [8]:

```
#Declaring and Assing a value to a variable
```

In [9]:

```
a = 2
print(a)
```

2

#The value of the variable can be changed after the program

In [10]:

```
a =6
print(a)
```

6

Naming Rules and keywords

#Naming rules

#Vasriables names should have a combination of letters in lowercase like(a to z) or uppercase (A to Z) or Digit (0 to 9) or an underscore(_).

In [11]:

```
A9a_ =1
```

In [13]:

```
#Create a name to make sense
```

In [14]:

```
wad = 100
price = 100
```

#If you want to create a variable have two name ,so you should give underscore to seperate them

In [17]:

```
Numbers_of_cats = 10
```

#Do not start a variable with any digit

In [19]:

```
1a = 100 # it is incorect
```

1.5.2 Keywords

#Keywords are the reserved number in python , They cannot be used as ordinary identifiers and must be spelled exactly. #There 35 keywords in python

In [21]:

```
def = 1 #When variable name is a keyword
```

In [23]:

```
import keyword
keyword.kwlist
```

Out[23]:

```
['False',
 'None',
 'True',
 '__peg_parser__',
 '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']
```

In [25]:

```
help('keywords')
```

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

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

1.6 Switching Values

In [26]:

```
a=1
b=6
print(a)
print(b)
```

```
1
6
```

a,b=b,a print(a) print(b)

1.7 Execution order in Python

#From top to bottom

In [ ]:

```
x=0
print(x+y)
y=1
```

1.8 Indentation

#A code Block(body of the function,loops etc)Starts with indentation and last indentation lines.the number of space to be indented is up to you,that must be consistant out through the block. #Indented makes the python code neat and clean .It also make the python code look similar and cosistent.

In [29]:

```
def funtion():
    a =1
    b = 4
def function2():
    a =3
    b=5
```

1.9 Global Variables And Local VAriables

#global variables are variavble declared in global scope(code block without indendation) #Local variables are variavble declared in local scope(code block with indendation)

In [30]:

```
a = 1 #Global variable
def function():# define the funtion
    b= 2# Define local function

c = 3
print(a+c)
```

```
4
```

In [31]:

```
print(a+f)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [31], in <cell line: 1>()
----> 1 print(a+f)

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

Standard O/I

#To read standard input,use input() Function #To write standard output,use print()function

In [32]:

```
s = ("input: ")
print(s)
```

```
input:
```

1.11 Python Basics Functionfunctions

#1.11.1 Help function #The python help function is ised to display the documentation of classes,modules,keyword,funcrtionsetc

In [33]:

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

1.11.2id()function

#id function return 'identity'of an object.This is na integer which is gurateed to be unique and constant for this object for this during lifetime

In [34]:

```
a =2
id(print)
```

Out[34]:

```
2295908415104
```

1.11.3 type()function

#type function returns the type of an object.

1.11.4 del()function

#del function delete the object

In [2]:

```
a = 2
del(a)
print(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [2], in <cell line: 3>()
      1 a = 2
      2 del(a)
----> 3 print(a)

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

1.11.5 len()function

#len function returns length(the numbers of items)an object

In [3]:

```
len('hello')
```

Out[3]:

```
5
```