1. What is Scripting?
Script is light weight programming
	a. Less Lines of Coding
	b. Interpreter based
	c. Line by Line Execution
	d. One line taking less memory

Script is loosely typed Programming
	a. No Data type declaration
	b. Implicit Declaration
	c. Dynamic Programming
	d. Lesser Lines of Logical statements

Script is Easy to Understand, Simple to implement:
	a. No Complex Syntax.
	b. No Header files req.
	c. No Curly bracket blocks
	d. Scripts are looks like general English language

2. Types of Scripts?
Generally Scripts are classified into the following two types:
1. Client Side Scripting Languages
A script which is get executes within the web browser is called Client Side Scripting Languages

Example:
HTML. CSS. JAVASCRIPT. jQuery. AngularJS. Angular. EmberJS. BackboneJS. LimeJS. PJS. D3JS....!!

2. Server Side Scripting Languages
A script which is get executes within the web server is called Client Side Scripting Languages

Example:
PYTHON ==> WSGI ==> Web Server Gateway Interface
JSP ==> Tomcat
PHP ==> Apache
.NET ==> IIS ==> Internet Information Services
NodeJS ==> Any WebServer (Server Side Java Script)

---

3. Programming
It is kind of logic development on a specific tech, to solve client business req. These techs are Two types:
1 ScriptingLanguages
	1. Less Lines of Coding
	2. Less Time Taking
	3. Less Errors
	4. More Quality
	5. More Productvity

2 Programming Languages
	1. More Lines of Coding
	2. More Time Taking
	3. More Errors

4. Less Quality
5. Less Productvity

4. Difference between scripting languages & programming languages

Scripting Languages
1. Light Weight Programming
2. Interpreter based
3. Easily Integrating with any tech.

Programming Languages:
1. Heavy Weight Programming
2. Compiler based
4. Difficult to integrate with other techs.

NOTE:
Python is not Scripting language+ Python is Not Programming Language
Python is Multi-Pardigm Tech.
Multi-Paradigm ==> It is the combination Scripting Languages Features+Programming Languages Features.....!!!

1. Python Features
a. Simple & Easy
b. Expressive Language
c. It is Dynamic Programming
d. No Complexity
e. Easier Syntax
f. Cross Platform Support
g. Huge List of Libs.!
i. NumPY
ii.SciPY
iii.Matplotlib
iv.Pandas
v.Math
vi.Random
vii.SKLearn (ScikitLearn)
viii.MLPY
ix.PyTorch
x.TensorFlow (TF)
xi.PyRenn
xii.PyAnn
xiii.NeuroScience
xiv.NeuroLab
xv. Seaborn
xvi.PyDoop
xvii.PySpark

--!!!@@@@@
h. Supports all Trendy Techs.

2. Python Installation
For installation of PYTHON S/W....!!!
https://www.python.org/downloads:
1. Click on windows ---> select latest version ==> Click on download
2. Double Click on exe file, it get install within 10 seconds successfully
3. Windows-7 OS on Not Rec.

4. Unix Linux OSs are having by default Python Existed
5. Mac/Android/iOS/Ubuntu/CentOS/Debian.....!!
6. After successful installation, u will get the following prompt:
IDLE ==> Intergrated Development Learning Enviroment
>>> It is Triple Chevron (Prompt of Python Programming)

```
>>> copyright
Displays copyright info.

>>> credits
Displaus Python Credit info.

>>> license()
Displays License Info.

>>> help()
Display help related any Python Statement or Command or Function or
Lib.....!!!
```

ColorName	Meaning
BlackColor	Standard Input
Blue	Standard Output
Red	Comment/Error
Green	String/Char/Documentation
Purple	Keyword/FunName/ReservedWord

```
5. print() function
>>> 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.
```


```
5. print() function
>>> 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.
```


```
Value Parameter:
```

This parameter contains all values related to Numbers or Strings or Special Characters also.

Example:

```
>>> print("Hello Welcome to Python Coding")
Hello Welcome to Python Coding
>>> print('Hello Welcome to Python Coding')
Hello Welcome to Python Coding
>>> print("""Hello Welcome to Python Coding""")
Hello Welcome to Python Coding
>>> print('''Hello Welcome to Python Coding''')
Hello Welcome to Python Coding
```

NOTE:
In Python Coding all quotes are represents strings only...!!
In Python No Char data type

1. Double Quotes indicates string
2. Single Quote indicates string
3. Triple Double Quotes indicates string documentation (Documentation one or more lines information)
4. Triple Single Quotes indicates char documentation (Documentation one or more lines information)

More Examples:

```
>>> print("Hey Welcome to 'Machine' Learning")
Hey Welcome to 'Machine' Learning
>>> print('Hey Welcome to "Machine" Learning')
Hey Welcome to "Machine" Learning
>>> print("""Hey Welcome to '''Machine''' Learning""")
Hey Welcome to '''Machine''' Learning
>>> print('''Hey Welcome to """Machine""" Learning''')
Hey Welcome to """Machine""" Learning
```

Nested Quotes: Quote inside another quote:

```
>>> print("Hello Welcome to "Python" Coding")
SyntaxError: invalid syntax
>>> print('Hello Welcome to 'Python' Coding')
SyntaxError: invalid syntax
>>> print("Hello Welcome to """Python""" Coding")
SyntaxError: invalid syntax
>>> print('''Hello Welcome to '''Python''' Coding''')
SyntaxError: invalid syntax
```

NOTE:
Inner Quotes required escape sequence characters. These are special characters. Using these characters we can do different operations in the logical coding...!!

Example:

```
>>> print("Hello Welcome to \"Python\" Coding")
Hello Welcome to "Python" Coding
>>> print('Hello Welcome to \'Python\' Coding')
Hello Welcome to 'Python' Coding
>>> print("Hello Welcome to \"""Python\""" Coding")
Hello Welcome to "Python" Coding
```

```
>>> print('''Hello Welcome to \'''Python\''' Coding''')
Hello Welcome to '''Python''' Coding

NOTE:
Back slash charcter is for join lines and separate characters...!!!

Join Lines using Escape Sequence Chars:
>>> print("Hello Welcome to

SyntaxError: EOL while scanning string literal
>>> print('Hello Welcome to

SyntaxError: EOL while scanning string literal
>>> print("Hello Welcome to \
Python Coding")
Hello Welcome to Python Coding
>>> print('Hello Welcome to \
Python Coding')
Hello Welcome to Python Coding

>>> print("Welcome to Python Coding")
Welcome to Python Coding
>>> print("Hello", "Welcome", "to Python Coding")
Hello Welcome to Python Coding
>>> print(123456789)
123456789
>>> print(1,2,3,4,5,6,7,8,9)
1 2 3 4 5 6 7 8 9
>>> print('1,2,3,4,5,6,7,8,9')
1,2,3,4,5,6,7,8,9
>>> print('1,2,3,4',5,6,7,8,9)
1,2,3,4 5 6 7 8 9
>>> print('"Hello", "Welcome", "to Python Coding"')
"Hello", "Welcome", "to Python Coding"
>>>

b. Sep
>>> print("Hello Welcome to Python")
Hello Welcome to Python
>>> print("Hello", "Welcome to Python")
Hello Welcome to Python
>>> print("Hello", "Welcome to Python",sep="*")
Hello*Welcome to Python
>>> print("Hello", "Welcome", "to Python",sep="*")
Hello*Welcome*to Python
>>> print("Hello", "Welcome", "to Python",sep="__")
Hello__Welcome__to Python
>>> print("Hello", "Welcome", "to Python",sep="**")
Hello**Welcome**to Python
>>> print(1,2,3,4,5,6)
1 2 3 4 5 6
>>> print(1,2,3,4,5,6,sep="")
123456
>>> print(1,2,3,4,5,6,sep="\t")
1       2       3       4       5       6
>>> print(1,2,3,4,5,6,sep="\n")
1
```

```
2
3
4
5
6
>>> print(1,2,3,4,5,6,sep="^^")
1^^2^^3^^4^^5^^6
>>> print("1234567",sep="__")
1234567
>>> print("1,2,3,4,5,6,7",sep="__")
1,2,3,4,5,6,7
>>> print(1,2,3,4,5,6,7,sep="__")
1__2__3__4__5__6__7
>>>
```

c. End

```
>>> print(1)print(2)
SyntaxError: invalid syntax
>>> print(1);print(2)
1
2
>>> print(1)
1
>>> print(2)
2
>>> print(1,end="");print(2)
12
>>> print(1,end="--");print(2)
1--2
>>> print(1,2,3,4,5,sep=",",end="--");print(200)
1,2,3,4,5--200
>>> print(1,2,3,4,5,sep=",",end="--");print(200,end="::")
1,2,3,4,5--200::
>>> print(1,2,3,4,5,sep=",",end="--");print(200,sep='{}',end="::")
1,2,3,4,5--200::
>>> print(1,2,3,4,5,sep=",",end="--
");print(200,300,400,500,sep='{}',end="::")
1,2,3,4,5--200{}300{}400{}500::
>>>
```

d. File
e. Flush

I will cover these two parameters in Adv Python Coding....!!

2. input() function

```
>>> help(input)
Help on built-in function input in module builtins:

input(prompt=None, /)
    Read a string from standard input.  The trailing newline is
stripped.

    The prompt string, if given, is printed to standard output without
a
    trailing newline before reading input.
```

```
    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise
EOFError.
    On *nix systems, readline is used if available.

>>> x=input("Enter Any Number: ")
Enter Any Number: 12
>>> y=input("Enter Any Number: ")
Enter Any Number: 3
>>> x
'12'
>>> y
'3'
>>> x+y
'123'
>>> a=input("Enter Any String: ")
Enter Any String: Hello
>>> b=input("Enter Any String: ")
Enter Any String: Sir
>>> a
'Hello'
>>> b
'Sir'
>>> a+b
'HelloSir'
>>> int(x)+int(y)
15
>>>

>>> x=10
>>> y=20
>>> x+y
30
>>> a=input("Enter Any Number: ")
Enter Any Number: 10
>>> b=input("Enter Any Number: ")
Enter Any Number: 20
>>> a+b
'1020'
>>> a
'10'
>>> b
'20'
>>> p="Hello"
>>> q="Sir"
>>> p+q
'HelloSir'
>>> p+" "+q
'Hello Sir'
```