# Python Syntax

- Execute Python syntax

In [4]:
```
print("hello, World")
```

```
hello, World
```

In [5]:
```
print("hello world")
```

```
hello world
```

- Python Indention

In [6]:
```
if 5>2:
    print("Five is greater than two!")
```

```
Five is greater than two!
```

In [21]:
```
n =input("Enter a 1st number")
x =input("Enter a 2nd number")
if n!=x:
    print(f"Number {n} is not equal to {x} :")

else:
    print(f"Number {n} is equal to {x} :")
```

```
Number 2 is not equal to 4 :
```

In [16]:
```
if 5>2:
print("this is true")
```

```
  Cell In[16], line 2
    print("this is true")
    ^
IndentationError: expected an indented block after 'if' statement on line 1
```

In [20]:
```
n =input("Enter a 1st number")
x =input("Enter a 2st number")

if n > x:
 print("1st is greater than 2nd!")
if x > n:
        print("1st is less than 2nd!")
```

```
1st is less than 2nd!
```

In [22]:
```
if 5 > 2:
 print("Five is greater than two!")
        print("Five is greater than two!")
```

```
  Cell In[22], line 3
    print("Five is greater than two!")
    ^
IndentationError: unexpected indent
```

# Python Variable

In [24]:
```
x = 5
y = "Hello, World!"
print(x)
print(y)
```

```
5
Hello, World!
```

- comment

In [25]:
```
#This is a comment . the comment start with hash (#)
print("Hello, World!")
```

```
Hello, World!
```

In [26]:
```
print("Hello, World!") #This is a comment
```

```
Hello, World!
```

- Multi line comment

In [27]:
```
# this is a comment
# written in
# More than one line
print("hello, World")
```

```
hello, World
```

In [28]:
```
''' This is a comment
written in
more than just on line
'''
print('Hello, world')
```

```
Hello, world
```

In [29]:
```
""" This is the multi line comment
written in
more than just one line
"""
print('hello, World')
```

```
hello, World
```

# Python Variable

- Creating variable

In [30]:
```
x = 5
y = "John"
print(x)
print(y)
```

```
5
John
```

In [31]:
```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

```
Sally
```

- casting

In [33]:
```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
print(x)
print(y)
print(z)
```

```
3
3
3.0
```

- Get the type

In [34]:
```
x = 5
y = "John"
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'str'>
```

- single or double quotes

In [35]:
```
x = "John"
# is the same as
x = 'John'
```

In [36]:
```
y = "swapnil "
y = 'swapnil'
```

- Case sensitive

In [39]:
```
a = 4
A = "Sally"
#A will not overwrite a
print(a)
print(A)
```

```
4
Sally
```

# Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _ )
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

In [40]:
```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

In [42]:
```
# Legal VAriable
print(myvar)
print(my_var)
print(_my_var)
print(myVar)
print(MYVAR)
print(myvar2)
```

```
John
John
John
John
John
John
```

In [43]:
```
#Illegal variable names :
2myvar = "john"
my-var = "john"
my var = "john"
```

```
  Cell In[43], line 2
    2myvar = "john"
    ^
SyntaxError: invalid decimal literal
```

# Multi Words Variable Names

- Camel case

In [45]:
```
myVariableName = "john"
print(myVariableName)
```

```
john
```

- Pascal Case

In [46]:
```
MyVariableName = "john"
print(MyVariableName)
```

```
john
```

- Snake Case

In [48]:
```
my_variable_name = "swapnil"
print(my_variable_name)
```

```
swapnil
```

# Many Values to multiple variable

In [49]:
```
x , y, z = 'Orange','Banana','Cherry'
print(x)
print(y)
print(z)
```

```
Orange
Banana
Cherry
```

- One Value to Multiple Variable

In [50]:
```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

```
Orange
Orange
Orange
```

- Unpack a Collection

In [51]:
```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

```
apple
banana
cherry
```

In [52]:
```
numbers = [5,6,7]
x,y,z = numbers
print(x)
print(y)
print(z)
```

```
5
6
7
```

# Output Variable

In [53]:
```
x = "Python is aswome "
print(x)
```

```
Python is aswome
```

In [54]:
```
# In the print() function, you output multiple variables, separated by a comma :
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

```
Python is awesome
```

In [55]:
```
# You can also use the + operator to output multiple variables:
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

```
Python is awesome
```

In [57]:
```
# For numbers, the + character works as a mathematical operator:
x = 5
y = 7
print(x + y) # here two number is added becausse number is int value
```

```
12
```

In [58]:
```
# In the print() function, when you try to combine a string and a number with the + operator, Python will
x = 5
y = "swapnil "
print(x + y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[58], line 4
      2 x = 5
      3 y = "swapnil "
----> 4 print(x + y)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [59]:
```
# The best way to output multiple variables in the print() function is to separate them with commas, whic
x = 5
y = "swapnil "
print(x,y)
```

```
5 swapnil
```

# Global Variables

Variables that are created outside of a function (as in all of the examples in the previous pages) are known as global variables.

Global variables can be used by everyone, both inside of functions and

In [60]:
```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

```
Python is awesome
```

In [62]:
```
x = "awesome" # This is  a globle variable

def myfunc():
  x = "fantastic" # Local variable
  print("Python is " + x)
```

```
myfunc()

print("Python is " + x)
```

```
Python is fantastic
Python is awesome
```

# The globle Keyword

In [63]:
```
# If you use the global keyword, the variable belongs to the global scope:
def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

```
Python is fantastic
```

In [64]:
```
# To change the value of a global variable inside a function, refer to the variable by using the global k
x = "awesome"

def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

```
Python is fantastic
```

# Variable Exercises

- Variable

In [72]:
```
x = 5
print(x)
```

```
5
```

In [73]:
```
x = "swapnil "
# is the same as
x = 'swapnil'
# True
```

In [74]:
```
# case- sensitive
a = 5
# is the same as
A = 5
# False
```

In [75]:
```
# Select the correct functions to print the data type of a variable:
print(type(myvar))
```

```
<class 'str'>
```

In [76]:
```
carname = "Volvo"
```

In [77]:
```
x = 50
```

- Variable Name

In [78]:
```
my_var = 29
Myvar =29
_myvar = 29
```

- Multiple Variable

In [79]:
```
x,y,z = 3,4,5
print(x)
print(y)
print(z)
```

```
3
4
5
```

In [80]:
```
x = y = z = "swapnil"
print(x)
print(y)
print(z)
```

```
swapnil
swapnil
swapnil
```

In [81]:
```
x,y,z = 'swapnil','kumar','mishra'
print(x)
print(y)
print(z)
```

```
swapnil
kumar
mishra
```

In [82]:
```
fruites = ["Apple","Orange","Guava"]
x, y,z = fruites
print(x)
print(y)
print(z)
```

```
Apple
Orange
Guava
```

In [83]:
```
fruites = ["Apple","Orange","Guava"]
x, y,z = fruites
print(x)
```

```
Apple
```

In [84]:
```
numbers = [ 5,87,77,79,35]
x,y,z,a,b = numbers
print(x)
print(y)
print(z)
print(a)
print(b)
```

```
5
87
77
79
35
```

- Output Variable

In [86]:
```
print('hello', 'world')
```

```
hello world
```

In [87]:
```
a = 'Hello'
b = 'World'
print(a+b)
```

```
HelloWorld
```

In [88]:
```
a = 4
b = 9
print(a+b)
```

- Globle variable

In [89]:
```
x = 'awesome'
def myfunc():
    x = 'fantastic'

myfunc()
print('Python is ' + x)
```

```
Python is awesome
```

In [90]:
```
def myfunc():
    global x
    x = "fantastic"
```

In [91]:
```
x = 'awesome'
def myfunc():
  global x
  x = 'fantastic'
myfunc()
print('Python is ' + x)
```

```
Python is fantastic
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: