In [3]:
```
# statement 1
print('Hello')

# statement 2
x = 20

# statement 3
print(x)
```

```
Hello
20
```

In [4]:
```
# two statements in a single
l = 10; b = 5

# statement 3
print('Area of rectangle:', l * b)

# Output Area of rectangle: 50
```

```
Area of rectangle: 50
```

In [5]:
```
addition = 10 + 20 + \
           30 + 40 + \
           50 + 60 + 70
print(addition)
# Output: 280
```

```
280
```

In [6]:
```
addition = (10 + 20 +
            30 + 40 +
            50 + 60 + 70)
print(addition)
# Output: 280
```

```
280
```

In [7]:
```python
# list of strings
names = ['Emma',
         'Kelly',
         'Jessa']
print(names)

# dictionary name as a key and mark as a value
# string:int
students = {'Emma': 70,
            'Kelly': 65,
            'Jessa': 75}
print(students)
```

```
['Emma', 'Kelly', 'Jessa']
{'Emma': 70, 'Kelly': 65, 'Jessa': 75}
```

In [8]:
```python
x = 5
# right hand side of = is a expression statement

# x = x + 10 is a complete statement
x = x + 10
```

In [9]:
```python
# create a function
def fun1(arg):
    pass  # a function that does nothing (yet)
```

In [10]:
```python
x = 10
y = 30
print(x, y)

# delete x and y
del x, y

# try to access it
print(x, y)
```

```
10 30

---------------------------------------------------------------------
------
NameError                                 Traceback (most recent call
last)
Cell In[10], line 9
      6 del x, y
      8 # try to access it
----> 9 print(x, y)

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

In [11]:

```
# Define a function
# function acceptts two numbers and return their sum
def addition(num1, num2):
    return num1 + num2  # return the sum of two numbers

# result is the return value
result = addition(10, 20)
print(result)
```

```
30
```

In [12]:

```
import datetime

# get current datetime
now = datetime.datetime.now()
print(now)
```

```
2023-10-07 02:51:05.511152
```

In [1]:

```
x = 10
y = 20

# adding two numbers
z = x + y
print('Sum:', z)

# Output 30
```

```
Sum: 30
```

In [2]:

```
# welcome message
print('Welcome to PYnative...')
```

```
Welcome to PYnative...
```

In [3]:

```
# This is a
# multiline
# comment
print('Welcome to PYnative...')
```

```
Welcome to PYnative...
```

In [4]:
```python
# Returns welcome message for a customer by customer name and location
# param name - Name of the customer
# param region - location
# return - Welcome message

def greet(name, region):
    message = get_message(region)
    return message + " " + name


# Returns welcome message by location
# param region - location
def get_message(region):
    if (region == 'USA'):
        return 'Hello'
    elif (region == 'India'):
        return 'Namaste'

print(greet('Jessa', 'USA'))
```

```
Hello Jessa
```

In [5]:
```python
# Returns welcome message for a customer by customer name and location
# param name - Name of the customer
# param region - location
# return - Welcome message

def greet(name, region):
    message = get_message(region)
    return message + " " + name
```

In [6]:
```python
def bonus(salary):
    """Calculate the bonus 10% of a salary ."""
    return salary * 10 / 100
```

In [7]:
```python
def greet(name, region):
    # below code is comment for testing
    # message = get_message(region)
    message= 'Hello'
    return message + " " + name

def get_message(region):
    if (region == 'USA'):
        return 'Hello'
    elif (region == 'India'):
        return 'Namaste'

print(greet('Jessa', 'USA'))
```

```
Hello Jessa
```

In [8]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'b
reak', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lamb
da', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'whil
e', 'with', 'yield']
```

In [9]:
```
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 [10]:
```
print(help('if'))
```

```
The "if" statement
******************

The "if" statement is used for conditional execution:

   if_stmt ::= "if" assignment_expression ":" suite
               ("elif" assignment_expression ":" suite)*
               ["else" ":" suite]

It selects exactly one of the suites by evaluating the expressions on
e
by one until one is found to be true (see section Boolean operations
for the definition of true and false); then that suite is executed
(and no other part of the "if" statement is executed or evaluated).
If all expressions are false, the suite of the "else" clause, if
present, is executed.

Related help topics: TRUTHVALUE

None
```

In [11]:
```
import keyword

print(keyword.iskeyword('if'))
print(keyword.iskeyword('range'))
```

```
True
False
```

In [12]:
```
x = 25
y = 20

z = x > y
print(z)  # True
```

```
True
```

In [13]:
```
x = 10
y = 20

# and to combine to conditions
# both need to be true to execute if block
if x > 5 and y < 25:
    print(x + 5)

# or condition
# at least 1 need to be true to execute if block
if x > 5 or y < 100:
    print(x + 5)

# not condition
# condition must be false
if not x:
    print(x + 5)
```

```
15
15
```

In [14]:
```
# is keyword demo
x = 10
y = 11
z = 10
print(x is y) # it compare memory address of x and y
print(x is z) # it compare memory address of x and z
```

```
False
True
```

In [15]:
```
my_list = [11, 15, 21, 29, 50, 70]
number = 15
if number in my_list:
    print("number is present")
else:
    print("number is not present")
```

```
number is present
```

In [16]:
```python
x = 75
if x > 100:
    print('x is greater than 100')
elif x > 50:
    print('x is greater than 50 but less than 100')
else:
    print('x is less than 50')
```

```
x is greater than 50 but less than 100
```

In [17]:
```python
print('for loop to display first 5 numbers')
for i in range(5):
    print(i, end=' ')

print('while loop to display first 5 numbers')
n = 0
while n < 5:
    print(n, end=' ')
    n = n + 1
```

```
for loop to display first 5 numbers
0 1 2 3 4 while loop to display first 5 numbers
0 1 2 3 4
```

In [19]:
```python
# def keyword: create function
def addition(num1, num2):
    print('Sum is', num1 + num2)

# call function
addition(10, 20)
```

```
Sum is 30
```

In [20]:
```python
# pass keyword: create syntactically empty function
# code to add in future
def sub(num1, num2):
    pass
```

In [21]:
```
# create class
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def show(self):
        print(self.name, self.age)


# create object
s = Student('Jessa', 19)
# call method
s.show()
```

```
Jessa 19
```

In [22]:
```
# Opening file
with open('sample.txt', 'r', encoding='utf-8') as fp:
    # read sample.txt
    print(fp.read())
```

```
---------------------------------------------------------------------
------
FileNotFoundError                          Traceback (most recent call
last)
Cell In[22], line 2
      1 # Opening file
----> 2 with open('sample.txt', 'r', encoding='utf-8') as fp:
      3     # read sample.txt
      4     print(fp.read())

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:2
86, in _modified_open(file, *args, **kwargs)
    279 if file in {0, 1, 2}:
    280     raise ValueError(
    281         f"IPython won't let you open fd={file} by default "
    282         "as it is likely to crash IPython. If you know what y
ou are doing, "
    283         "you can use builtins' open."
    284     )
--> 286 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'sample.txt'
```

In [23]:
```
# Opening file
with open('sample.txt', 'r', encoding='utf-8') as fp:
    # read sample.txt
    print(fp.read())
```

```
---------------------------------------------------------------------
------
FileNotFoundError                         Traceback (most recent call
last)
Cell In[23], line 2
      1 # Opening file
----> 2 with open('sample.txt', 'r', encoding='utf-8') as fp:
      3     # read sample.txt
      4     print(fp.read())

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:2
86, in _modified_open(file, *args, **kwargs)
    279 if file in {0, 1, 2}:
    280     raise ValueError(
    281         f"IPython won't let you open fd={file} by default "
    282         "as it is likely to crash IPython. If you know what y
ou are doing, "
    283         "you can use builtins' open."
    284     )
--> 286 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'sample.txt'
```

In [24]:
```
# import only datetime class
from datetime import datetime

# get current datetime
now = datetime.now()
print(now)
```

```
2023-10-07 02:57:56.344015
```

In [25]:
```
def addition(num1, num2):
    return num1 + num2  # return sum of two number

# call function
print('Sum:', addition(10, 20))
```

```
Sum: 30
```

In [26]:
```python
price = 900  # Global variable

def test1():  # defining 1st function
    print("price in 1st function :", price)  # 900

def test2():  # defining 2nd function
    print("price in 2nd function :", price)  # 900

# call functions
test1()
test2()

# delete variable
del price
```

```
price in 1st function : 900
price in 2nd function : 900
```

In [27]:
```python
x = 10
y = 40
print(x + y)
# Output 50
```

```
50
```

In [28]:
```python
name = "Kelly"
surname = "Ault"
print(surname + " " + name)
# Output Ault Kelly
```

```
Ault Kelly
```

In [29]:
```python
x = 10
y = 40
print(y - x)
# Output 30
```

```
30
```

In [30]:
```python
x = 2
y = 4
z = 5
print(x * y)
# Output 8 (2*4)
print(x * y * z)
# Output 40 (2*4*5)
```

```
8
40
```

In [31]:
```
name = "Jessa"
print(name * 3)
# Output JessaJessaJessa
```

```
JessaJessaJessa
```

In [32]:
```
x = 2
y = 4
z = 8
print(y / x)
# Output 2.0
print(z / y / x)
# Output 1.0
# print(z / 0)  # error
```

```
2.0
1.0
```

In [33]:
```
x = 2
y = 4
z = 2.2

# normal division
print(y / x)
# Output 2.0

# floor division to get result as integer
print(y // x)
# Output 2

# normal division
print(y / z)  # 1.81

# floor division.
# Result as float because one argument is float
print(y // z)  # 1.0
```

```
2.0
2
1.8181818181818181
1.0
```

In [34]:
```
x = 15
y = 4

print(x % y)
# Output 3
```

```
3
```

In [35]:

```python
num = 2
# 2*2
print(num ** 2)
# Output 4

# 2*2*2
print(num ** 3)
# Output 8
```

```
4
8
```

In [36]:

```python
x = 10
y = 5
z = 2

# > Greater than
print(x > y)  # True
print(x > y > z)  # True

# < Less than
print(x < y)  # False
print(y < x)  # True

# Equal to
print(x == y)  # False
print(x == 10)  # True

# != Not Equal to
print(x != y)  # True
print(10 != x)  # False

# >= Greater than equal to
print(x >= y)  # True
print(10 >= x)  # True

# <= Less than equal to
print(x <= y)  # False
print(10 <= x)  # True
```

```
True
True
False
True
False
True
True
False
True
True
False
True
```

In [37]:

```
a = 4
b = 2

a += b
print(a)  # 6

a = 4
a -= 2
print(a)  # 2

a = 4
a *= 2
print(a)  # 8

a = 4
a /= 2
print(a)  # 2.0

a = 4
a **= 2
print(a)  # 16

a = 5
a %= 2
print(a)  # 1

a = 4
a //= 2
print(a)  # 2
```

```
6
2
8
2.0
16
1
2
```

In [38]:

```
print(True and False)  # False
# both are True
print(True and True)  # True
print(False and False)  # False
print(False and True)  # false

# actual use in code
a = 2
b = 4

# Logical and
if a > 0 and b > 0:
    # both conditions are true
    print(a * b)
else:
    print("Do nothing")
```

```
False
True
False
False
8
```

In [39]:

```
print(10 and 20) # 20
print(10 and 5)  # 5
print(100 and 300) # 300
```

```
20
5
300
```

In [40]:

```
print(True or False)  # True
print(True or True)  # True
print(False or False)  # false
print(False or True)  # True

# actual use in code
a = 2
b = 4

# Logical and
if a > 0 or b < 0:
    # at least one expression is true so conditions is true
    print(a + b)  # 6
else:
    print("Do nothing")
```

```
True
True
False
True
6
```

In [41]:
```python
print(10 or 20) # 10
print(10 or 5) # 10
print(100 or 300) # 100
```

```
10
10
100
```

In [42]:
```python
print(not False)  # True return complements result
print(not True)  # True return complements result

# actual use in code
a = True

# Logical not
if not a:
    # a is True so expression is False
    print(a)
else:
    print("Do nothing")
```

```
True
False
Do nothing
```

In [43]:
```python
print(not 10) # False. Non-zero value
print(not 1)  # True. Non-zero value
print(not 5)  # False. Non-zero value
print(not 0)  # True. zero value
```

```
False
False
False
True
```

In [44]:
```python
my_list = [11, 15, 21, 29, 50, 70]
number = 15
if number in my_list:
    print("number is present")
else:
    print("number is not present")
```

```
number is present
```

In [45]:
```python
my_tuple = (11, 15, 21, 29, 50, 70)
number = 35
if number not in my_tuple:
    print("number is not present")
else:
    print("number is present")
```

```
number is not present
```

In [46]:
```
x = 10
y = 11
z = 10
print(x is y) # it compare memory address of x and y
print(x is z) # it compare memory address of x and z
```
```
False
True
```

In [47]:
```
x = 10
y = 11
z = 10
print(x is not y) # it campare memory address of x and y
print(x is not z) # it campare memory address of x and z
```
```
True
False
```

In [48]:
```
a = 7
b = 4
c = 5
print(a & b)
print(a & c)
print(b & c)
```
```
4
5
4
```

In [49]:
```
a = 7
b = 4
c = 5
print(a | b)
print(a | c)
print(b | c)
```
```
7
7
5
```

In [50]:
```
a = 7
b = 4
c = 5
print(a ^ c)
print(b ^ c)
```
```
2
1
```

In [51]:
```
a = 7
b = 4
c = 3
print(~a, ~b, ~c)
# Output -8 -5 -4
```
```
-8 -5 -4
```
In [52]:
```
print(4 << 2)
# Output 16
print(5 << 3)
# Output 40
```
```
16
40
```
In [53]:
```
print(4 >> 2)
# Output
print(5 >> 2)
# Output
```
```
1
1
```
In [54]:
```
print((10 - 4) * 2 +(10+2))
# Output 24
```
```
24
```
In [ ]:
```

```