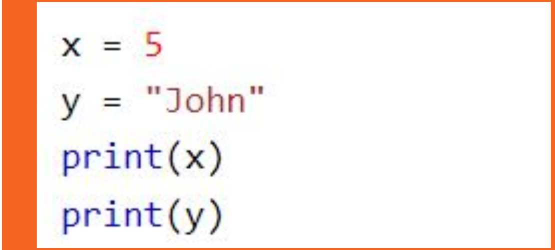
Python programming

Md Nakibul Islam

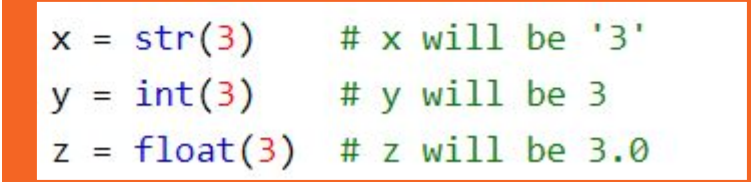
Print

```
Print(“Hello Bangladesh”)
```

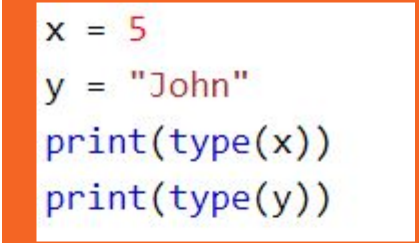
Variables



```
x = 5
y = "John"
print(x)
print(y)
```

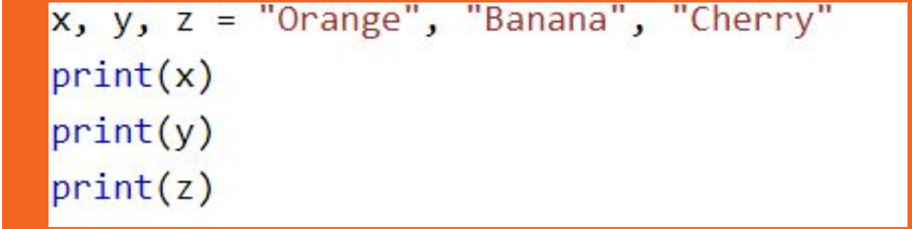


```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

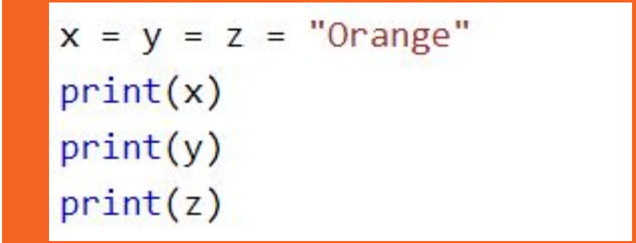


```
x = 5
y = "John"
print(type(x))
print(type(y))
```

Multiple Variables

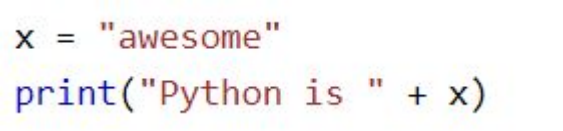


```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

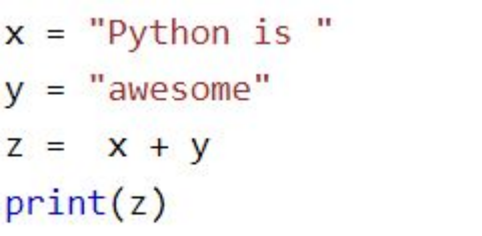


```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

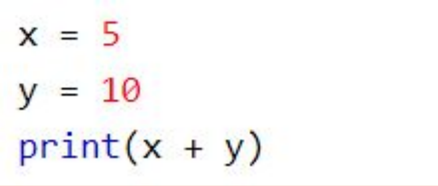
Output Variables



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

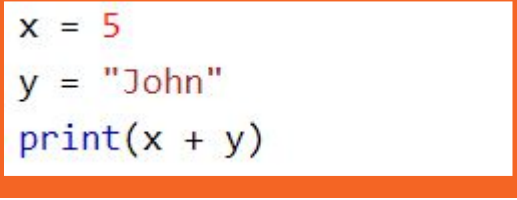


```
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```



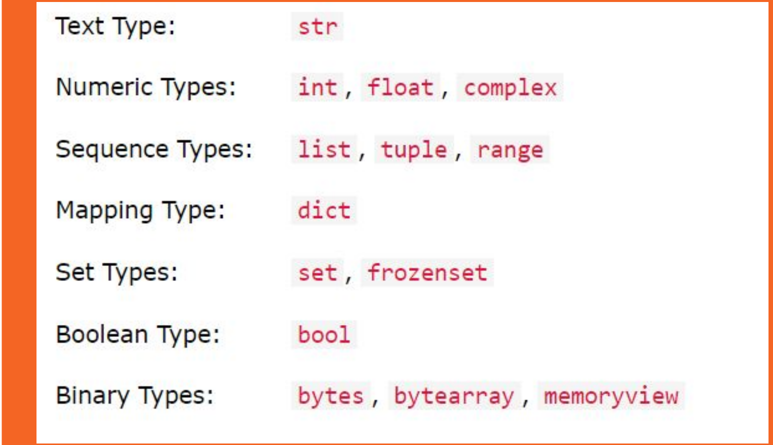
```
x = 5
y = 10
print(x + y)
```

Not work



```
x = 5
y = "John"
print(x + y)
```

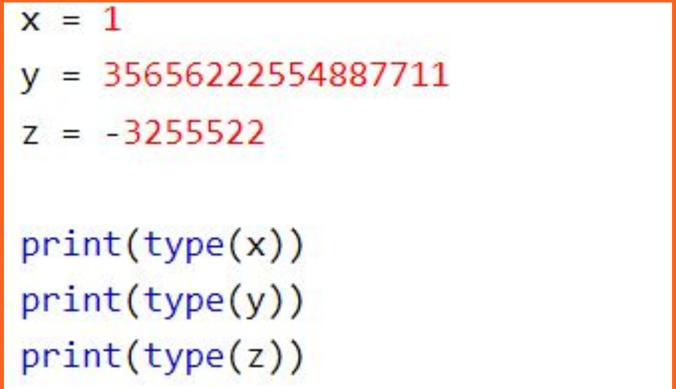
Data Types



Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

Data Types

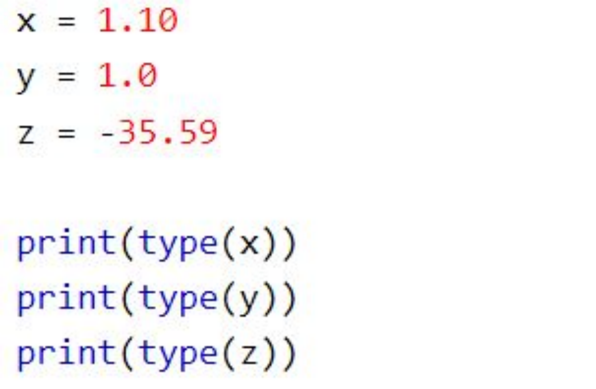
Int



```
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

Data Types

Float

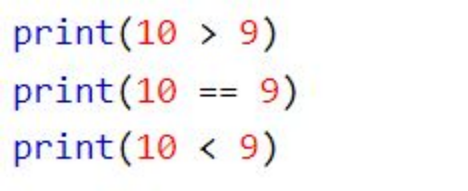


```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

Data Types

bool

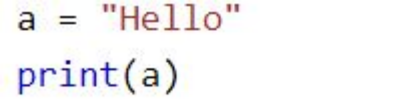
True
False



```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Data Types

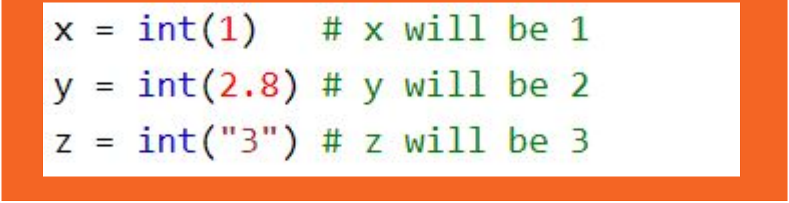
String



```
a = "Hello"
print(a)
```

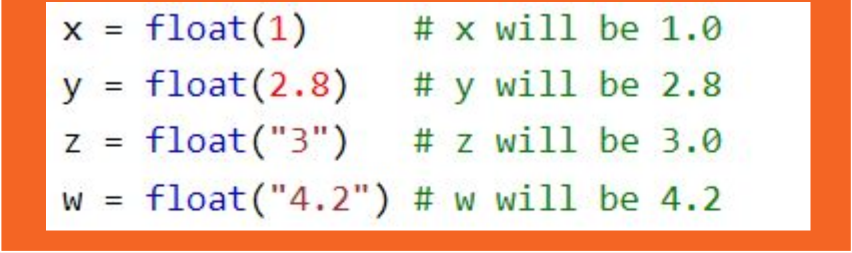
Types Casting

Int



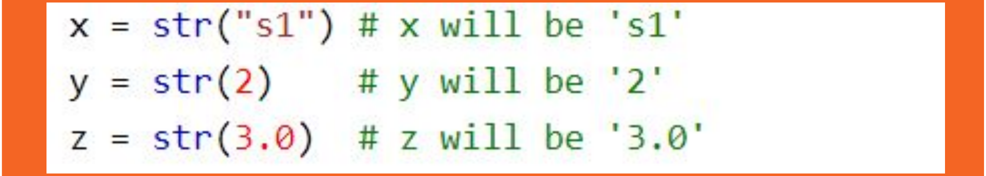
```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Float



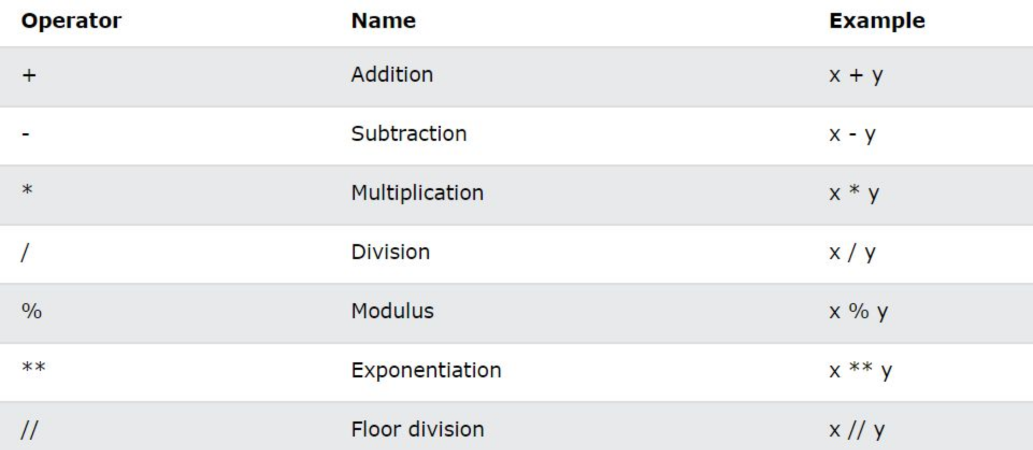
```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

String



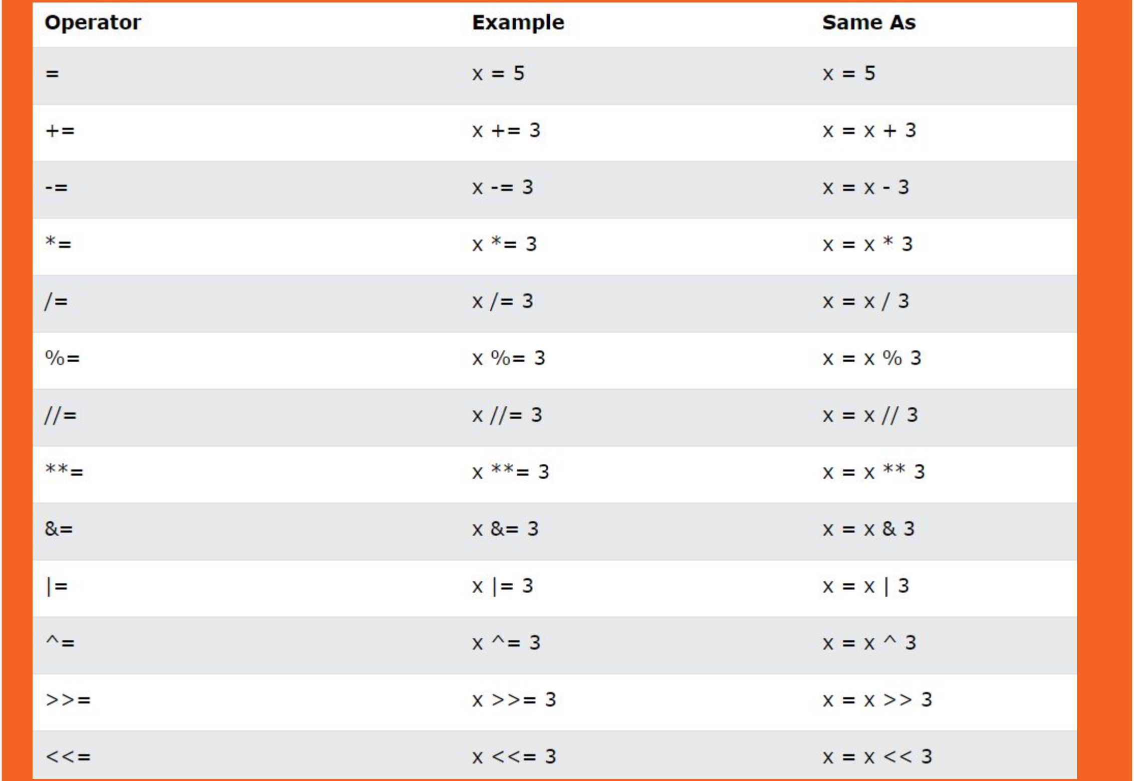
```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Arithmetic Operators



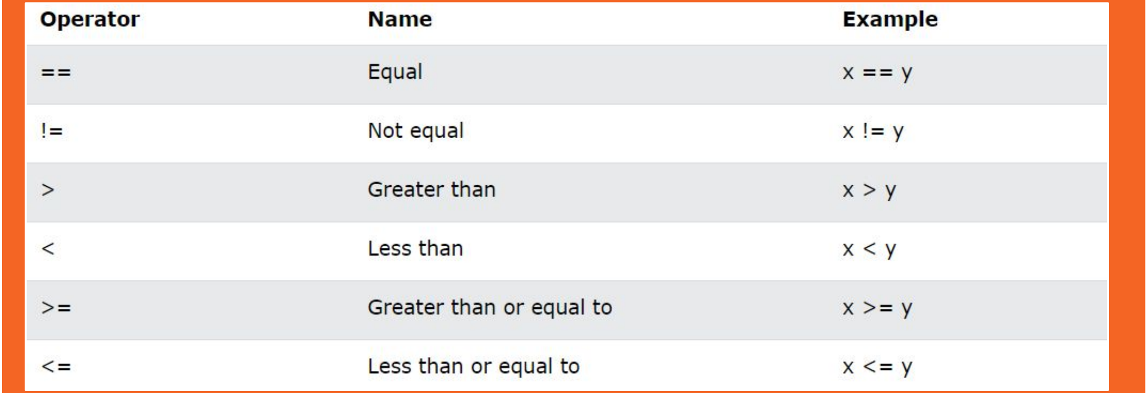
Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment Operators



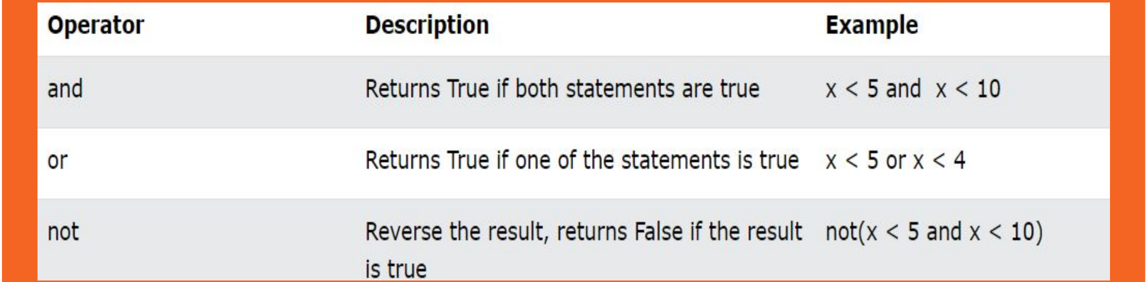
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Comparison Operators



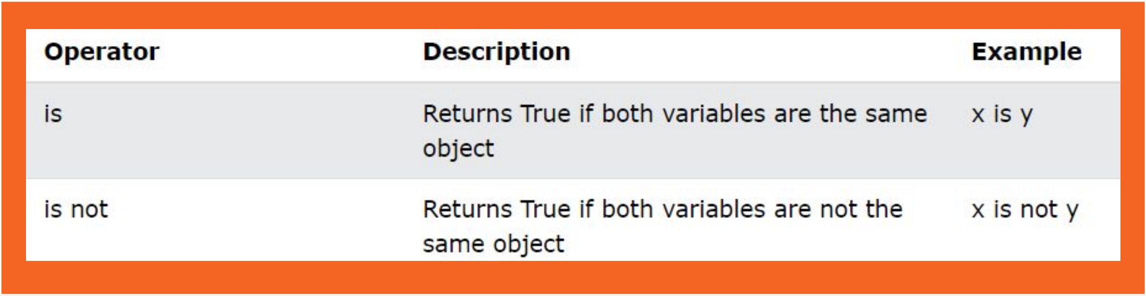
Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators



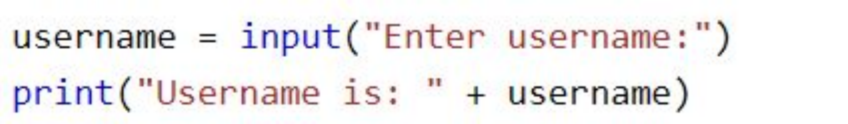
Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Identity Operators

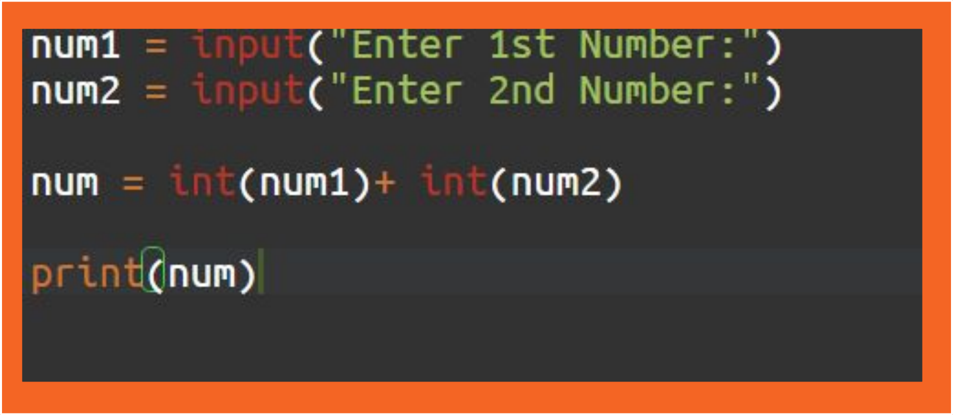


Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

User Input

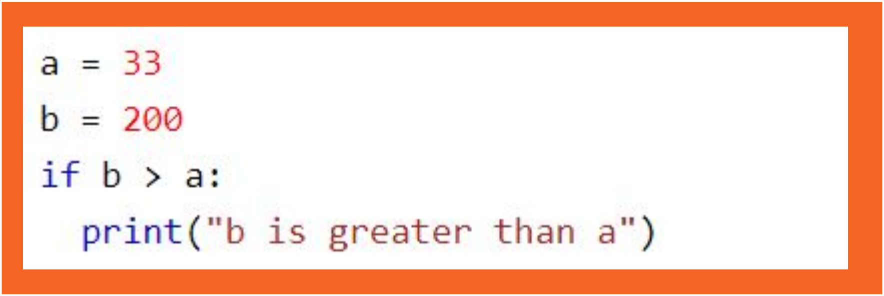


```
username = input("Enter username:")
print("Username is: " + username)
```

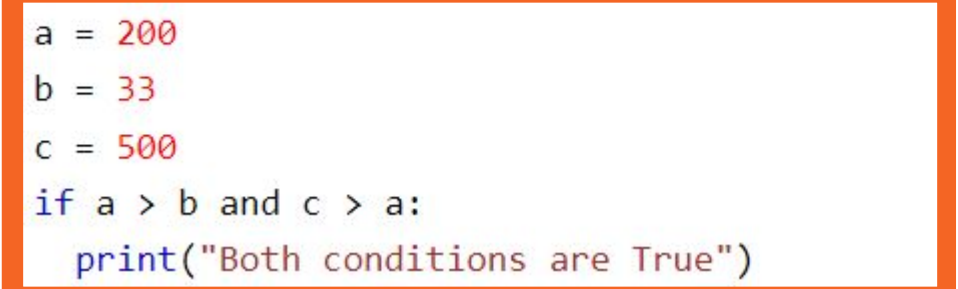


```
num1 = input("Enter 1st Number:")
num2 = input("Enter 2nd Number:")

num = int(num1)+ int(num2)

print(num)
```

if

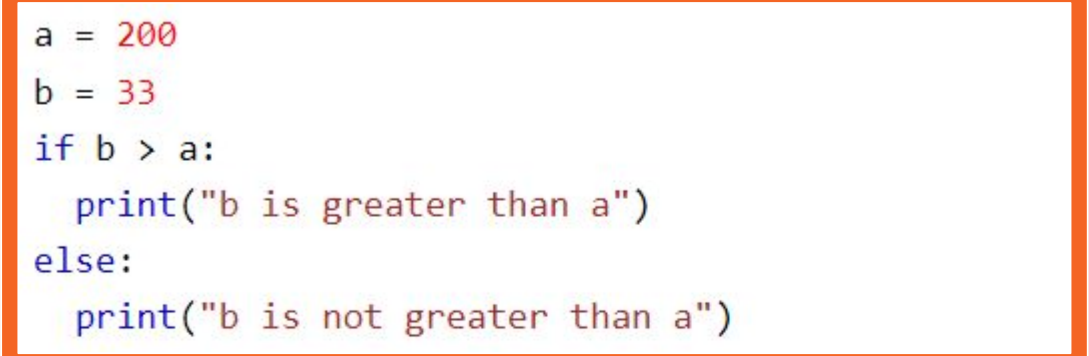


```python
a = 33
b = 200
if b > a:
  print("b is greater than a")
```



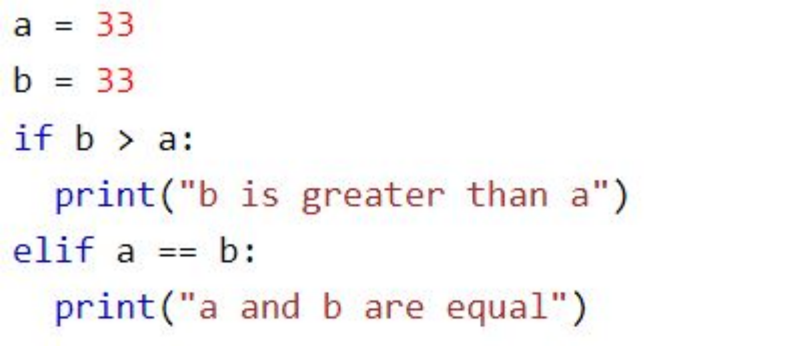
```python
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

else



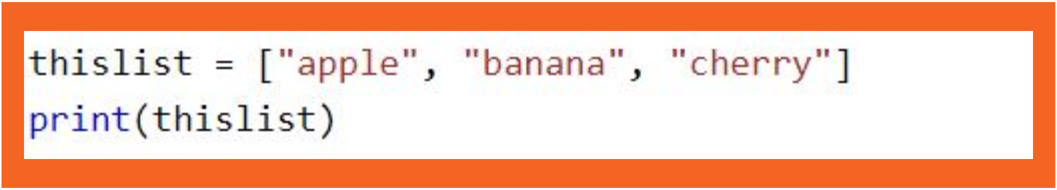
```python
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

elif

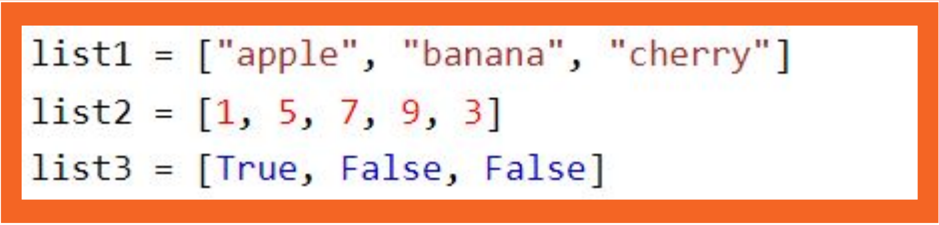


```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

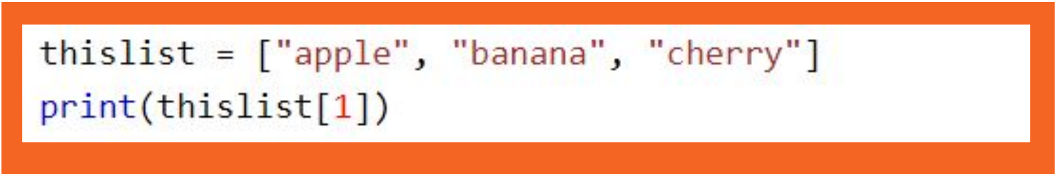
Lists



```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

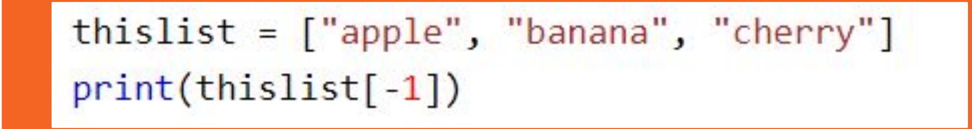


```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```



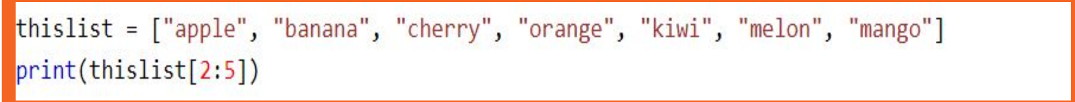
```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

Lists



```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

-1 refers to the last item, -2 refers to the second last item etc.

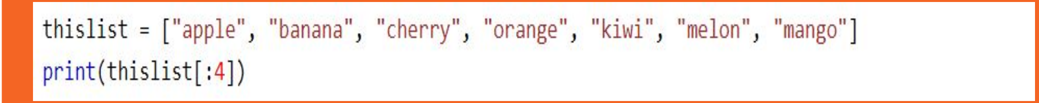


```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

#This will return the items from position 2 to 5.

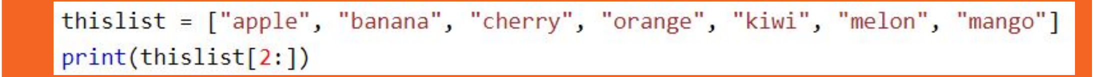
#Remember that the first item is position 0,
#and note that the item in position 5 is NOT included

Lists



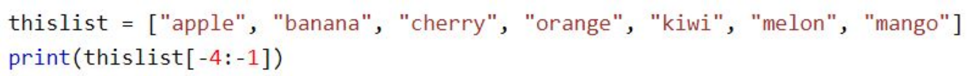
```python
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

#This will return the items from index 0 to index 4.



```python
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])
```

#This will return the items from index 2 to the end.

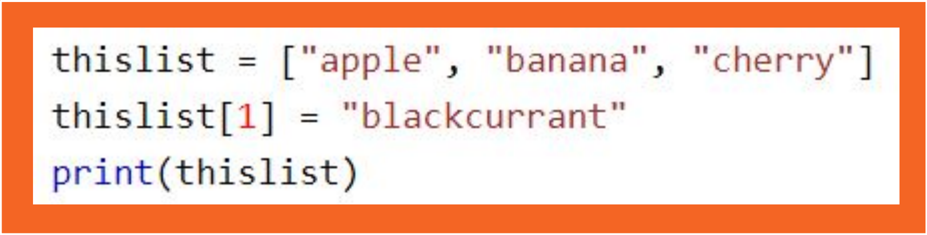


```python
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

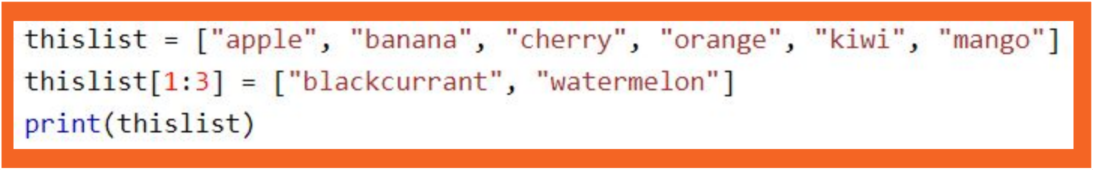
#Negative indexing means starting from the end of the list.

#This example returns the items from index -4 (included) to index -1 (excluded)

Change List items

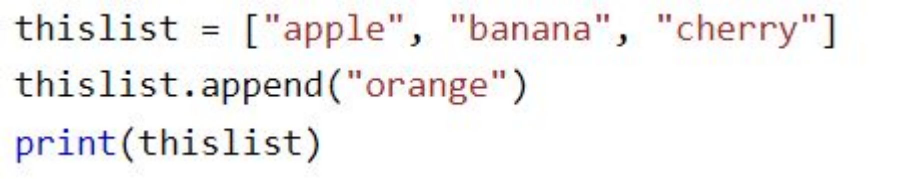


```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

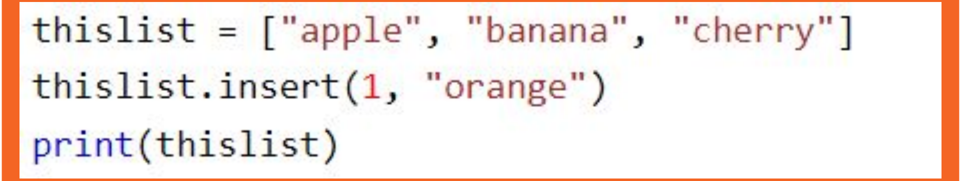


```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```

Add List items

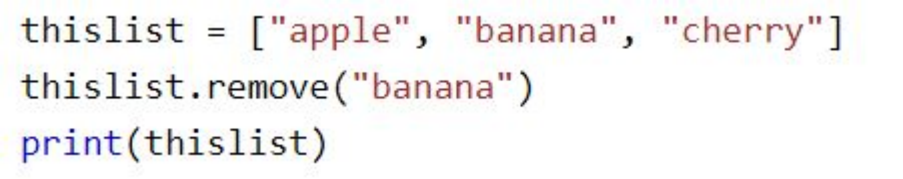


```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

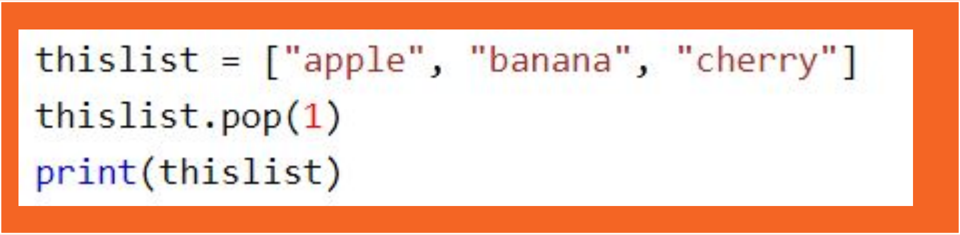


```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

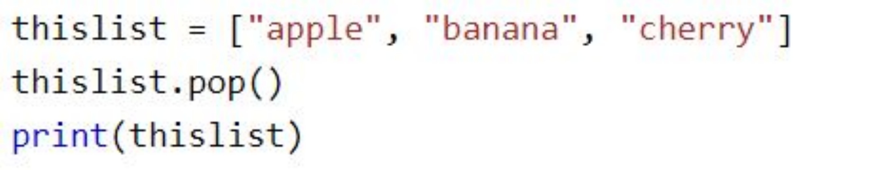
Remove List items



```python
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```



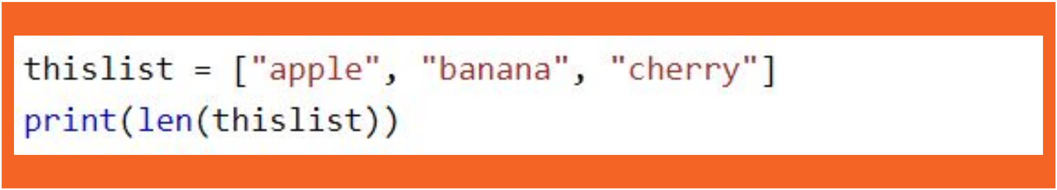
```python
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```



```python
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

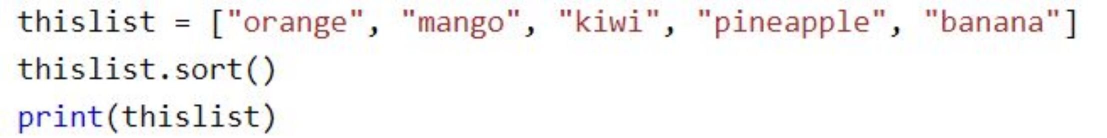
If you do not specify the index, the `pop()` method removes the last item

Lists Length

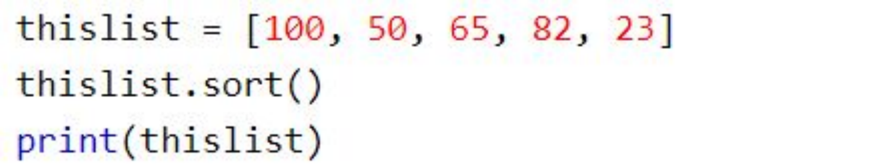


```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

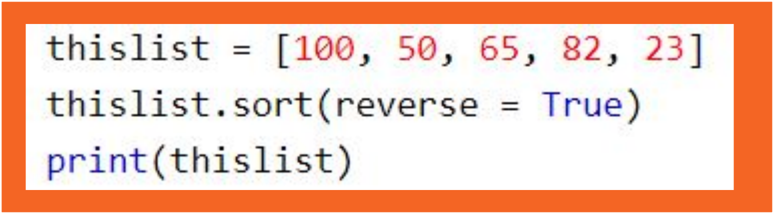
Lists Sort



```python
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

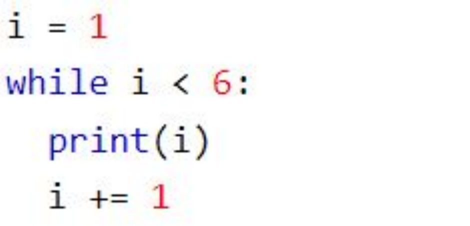


```python
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
```



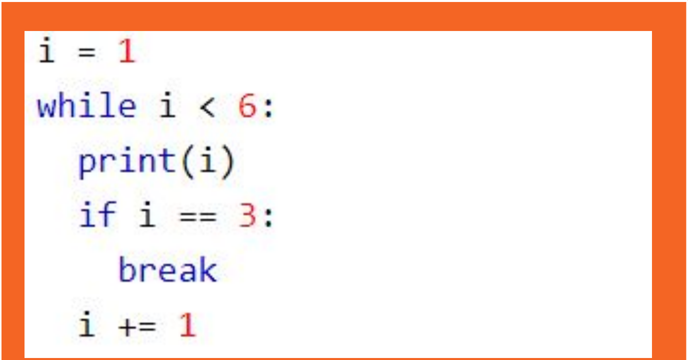
```python
thislist = [100, 50, 65, 82, 23]
thislist.sort(reverse = True)
print(thislist)
```

While loop



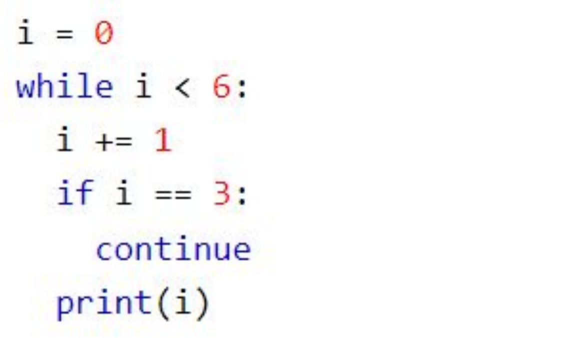
```
i = 1
while i < 6:
  print(i)
  i += 1
```

While loop break



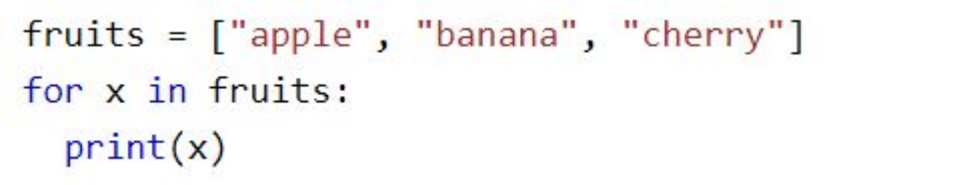
```
i = 1
while i < 6:
 print(i)
 if i == 3:
  break
 i += 1
```

While loop continue

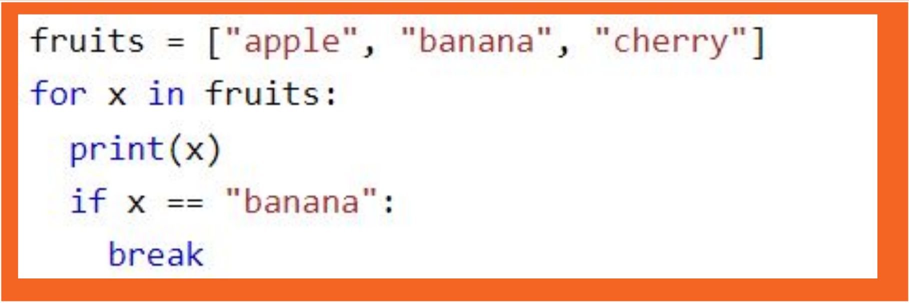


```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

For loop

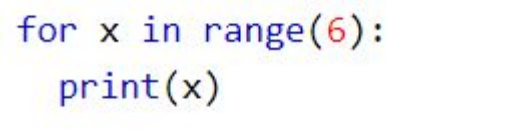


```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

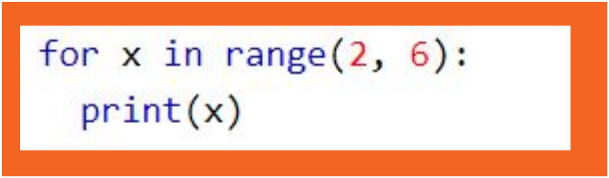


```python
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

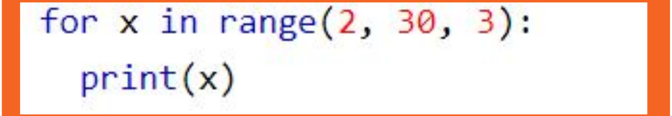
For loop
Range



```
for x in range(6):
  print(x)
```

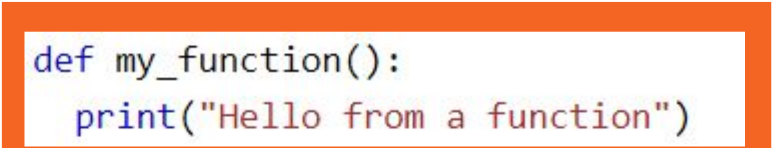


```
for x in range(2, 6):
  print(x)
```

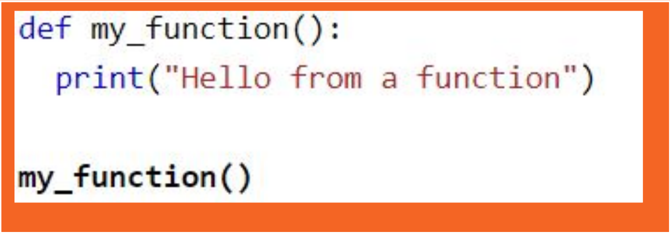


```
for x in range(2, 30, 3):
  print(x)
```

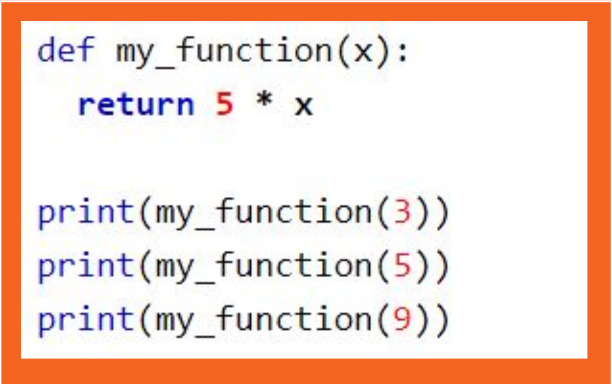
Functions



```
def my_function():
  print("Hello from a function")
```

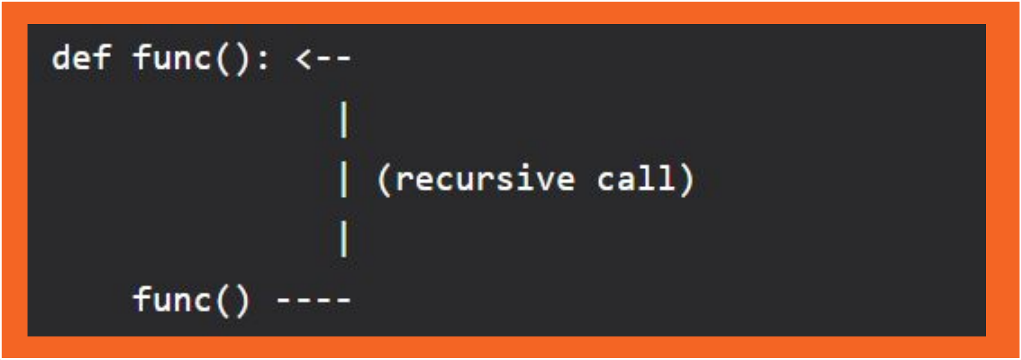


```
def my_function():
  print("Hello from a function")

my_function()
```



```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Recursion



```
def func(): <--
              |
              | (recursive call)
              |
    func() ----
```