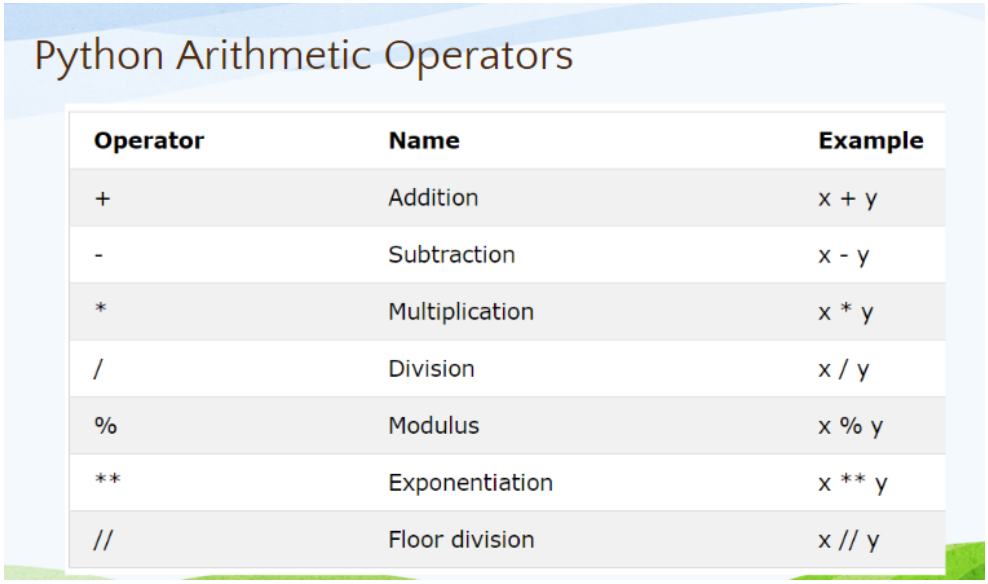
Python Review Cheat Sheet

Python 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

Python Variable Assignment:

```
a = 2
a, b = 2, 3
x, y = y, x
```

Using a variable as a counter

```
a = a + 2
a += 2
```

In [ ]:
```
a *= 2   a = a *2
```

In [ ]:
```

```

Print with Format

Placeholders:

- %d integer
- %f float
- %s string

Examples:

In [24]:
```
#Get Keyboard Input
name = input('please type your nameP: ')
age = int(input('How old are you? '))
print('%s\'s age is %d months'%(name, age))
```

```
please type your nameP: larry
How old are you? 144
larry's age is 144 months
```

In [18]:
```
pi = 3.1415926535
print("PI is %f" % pi)
print("PI is %.2f" %pi)
print("PI is %5.3f" % pi)
print("PI is %6.3f" % pi)
print("PI is %7.3f" % pi)

r = 5
area = r*r*pi
print("R is %d, Area is %.3f"% (r, area))
print("R is %d, Area is %7.3f"% (r, area))
```

```
PI is 3.141593
PI is 3.14
PI is 3.142
PI is  3.142
PI is   3.142
R is 5, Area is 78.540
R is 5, Area is  78.540
```

In [3]:
```
a = 123
b = 45678
print(a)
print(b)

# %5d means fill the number with space if it is less than 5 digits
print("%5d"%a)

# if a variable has more digits than %xd defined, print number as is (do noth
print("%3d"%b)
```

```
123
45678
  123
45678
```

Python String

Defined by single, double or triple quote marks:

```
name = "Frank"
name = 'Larry'
```

In [5]:
```
# use triple quote marks to include single or double quote marks in a string
name = ''' King's Avatar("A.K.A"全职高手)'''
print(name)
```

```
 King's Avatar("A.K.A"全职高手)
```

The whitespace

The term "whitespace" refers to characters that the computer is aware of, but are invisible to human.

```
\t
\n
"" or ''
```

In [26]:
```
print("Twinkle, twinkle, little star, \n\tHow I wonder what you are!")
```

```
Twinkle, twinkle, little star,
        How I wonder what you are!
```

String Escaping

String Escaping is to prevent certain characters being used as a part of the coding language, and have them be represented correctly in the string itself. Characters that need excaption: ", ', \

In [2]:
```
msg = "String escaption can be applied to character \", \', and \\"
print(msg)
```

```
String escaption can be applied to character ", ', and \
```

Useful String Methods

```
strip() remove space from the beginning and end of the string.

len() get string length

find() search string in a string
count() count the times a substring appears in the string

split() parse the string with a given seperator

upper() to upper case
lower() to lower case
title() capitalize the first character of each word
replace()  replace a substring with another string
```

```
in:
    when used with if: if "br" in "brother":
    when used with for: for c in "brother":
not in
```

In [11]:
```
str1 = " important  imformation:  "
print(len(str1))
print(len(str1.strip()))
```

```
26
23
```

In [2]:
```
song = "Twinkle, twinkle, little star, \n\tHow I wonder what you are!"
print(song.count('twin'))
```

```
1
```

String Operations

- Concatenate (join) strings: +
- Repeat: *

In [6]:
```
last_name = "Smith"
first_name = "John"
full_name = first_name + last_name
print(full_name)

star = "*"
stars = star*5
print(stars)
```

```
JohnSmith
*****
```

String is a special list

Access characters in the string using a position index:

In [1]:
```
msg = "Good Morning!"
m = msg[5]
print(m)
for c in msg:
    print(c, end=' ')
```

```
M
G o o d   M o r n i n g !
```

Reverse a String

String itself does not have reverse function.

In [11]:
```
msg = "nohtyP"

s = ''
for w in msg:
    s =  w + s
print(s)
```

```
Python
```

String Slice Operation

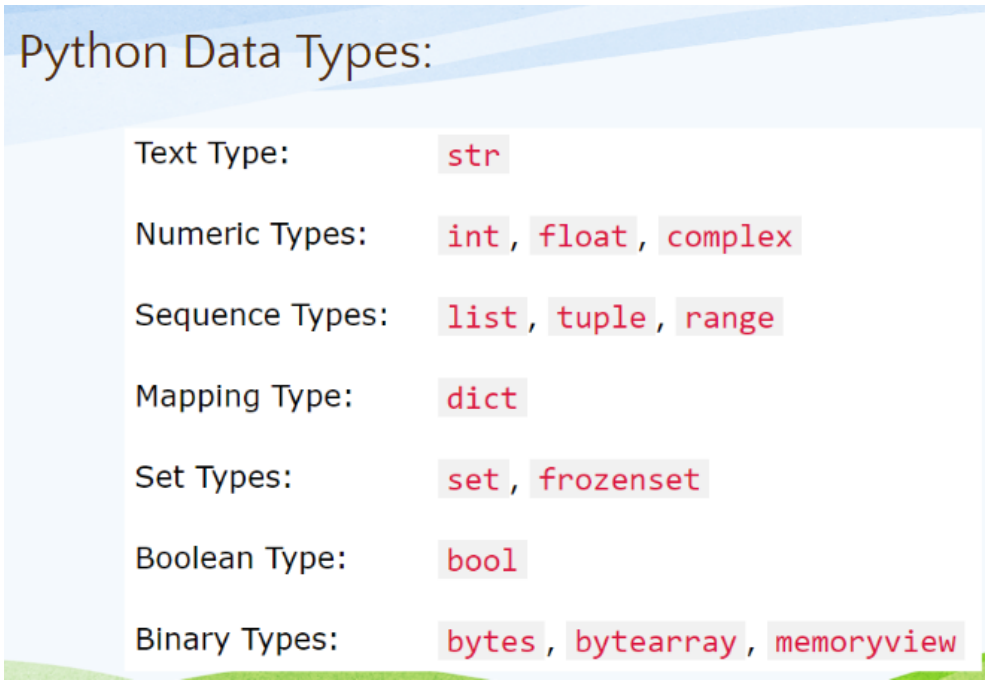
By using two numbers seperated by ':', define a starting index and an ending index, to extract a sub-string.

Note: the ending index is not included in the sub-string

In [10]:
```
# Slice Operation

msg = "Good-Morning!"
w = msg[0:4]
print(w)
print(msg[4])
```

```
Good
-
```

Python Data Types



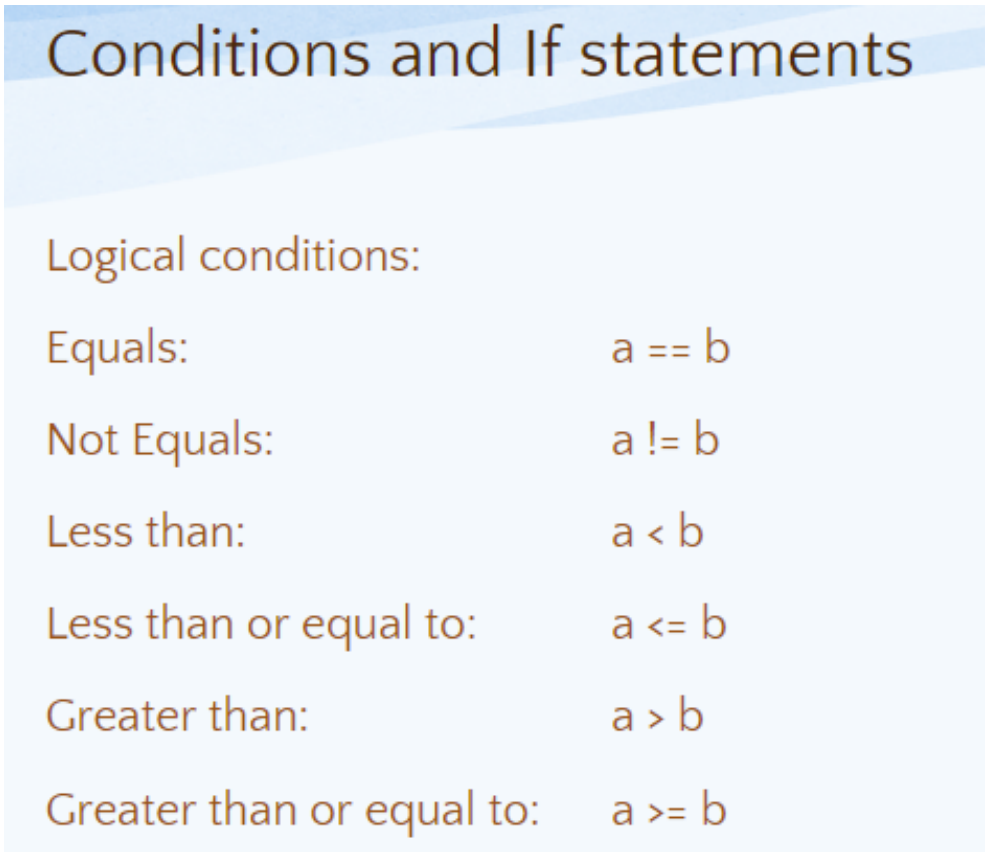
Data Types Functions

```
get the data type:
    type()

convert to data type:
    str(), int(), float()
```

In [13]:
```
bag = [123, 'name']
print(bag)

for item in bag:
    print(type(item))
```

```
[123, 'name']
<class 'int'>
<class 'str'>
```

Condition and if statement



```
if
elif
else

and
or
not
```

For Loop

For loop has two forms:

```
1) works with range()
for i in range(10):

2) works with a list/string
days = ['Monday', 'Tuesday']
for x in days:

text = "Python is easy"
for c in text:
```

for... else:

```
for i in range(10):

else
```

Continue and Break

Continue: jump to the next iteration of the loop

Break: exit the loop

The range() function

```
range(start, end, step)

range(5)
range(-1, 5, 1)
range(5, 0, -1)
```

While loop

If you use "while True:", to avoid infinite loop, make sure you setup a loop exit condition.

In [2]:
```
num = 1
while True:
    print('*'*num)
    num = num + 1
    # exit condition below
    if num > 5:
        break
```

```
*
**
***
****
*****
```

Python List

- len(): the length of the list
- max() and min(): maximum and minmum value in the list
- index(): find an item's location
- in: check existence
- append(): add more items from the back
- insert(): add item at a given position
- my_list = []: empty a list
- my_list.clear(): empty a list
- sort(): sort a list
- reverse(): reverse a list
- extend(): extend (join) two lists
- del: removing items from list
- pop(): removing from the back of a list
- range(): create a list with numbers
- enumerate(): return both index and item in the list, one by one

To get the details of the above functions, there are many online tutorials. Written tutorials:

- https://www.w3schools.com/python/python_lists.asp (https://www.w3schools.com/python/python_lists.asp)
- https://www.tutorialspoint.com/python/python_lists.htm (https://www.tutorialspoint.com/python/python_lists.htm)

Video tutorial

- https://youtu.be/ohCDWZgNIU0 (https://youtu.be/ohCDWZgNIU0)

Create a list and set the same initial value to every item

In [12]:
```
group = [True]*10
print(group)
```

```
[True, True, True, True, True, True, True, True, True, True]
```

Initial an empty list

In [ ]:
```
names = []
```