**Number**

* Integer: Integer(negative, zero and positive) numbers Example: ... -3, -2, -1, 0, 1, 2, 3 ...
* Float: Decimal number Example ... -3.5, -2.25, -1.0, 0.0, 1.1, 2.2, 3.5 ...
* Complex Example 1 + j, 2 + 4j

**String**

A collection of one or more characters under a single or double quote. If a string is more than one sentence then we use a triple quote.

#### Booleans

A boolean data type is either a True or False value. T and F should be always uppercase.

#### List

Python list is an ordered collection which allows to store different data type items. A list is similar to an array in JavaScript.

#### Dictionary

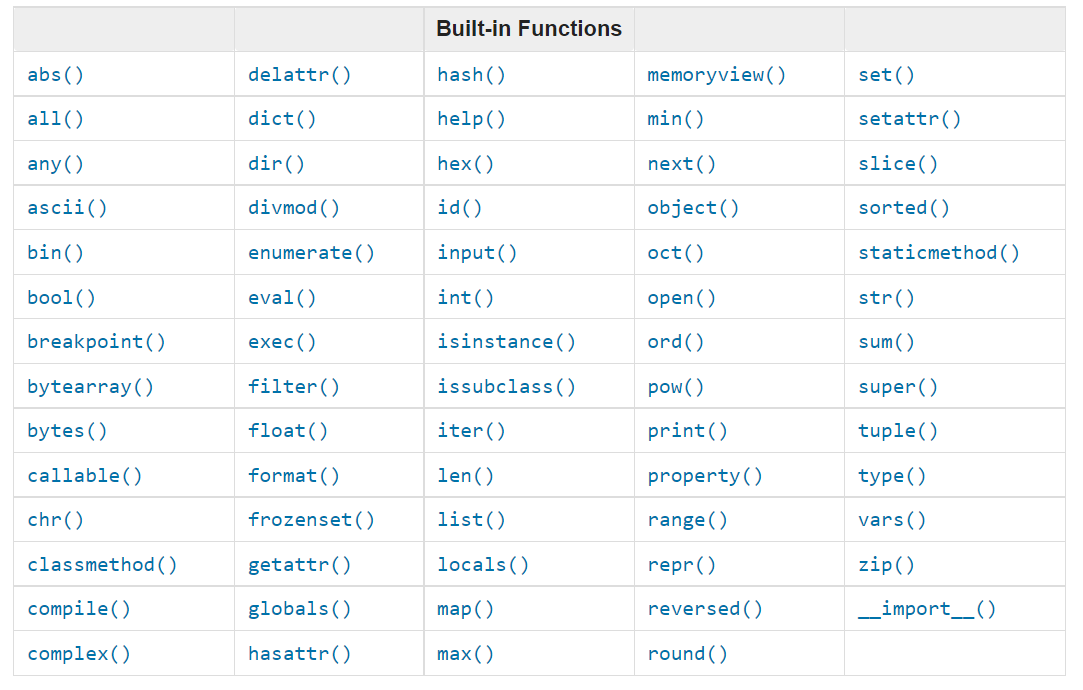
A Python dictionary object is an unordered collection of data in a key value pair format.

#### Tuple

A tuple is an ordered collection of different data types like list but tuples can not be modified once they are created. They are immutable.

#### Set

A set is a collection of data types similar to list and tuple. Unlike list and tuple, set is not an ordered collection of items. Like in Mathematics, set in Python stores only unique items.



## **Variables**

### Declaring Multiple Variable in a Line

Multiple variables can also be declared in one line:

first\_name, last\_name, country, age, is\_married = 'Asabeneh', 'Yetayeh', 'Helsink', 250, True

first\_name = input('What is your name: ')

print(first\_name)

print(type(first\_name)) # str

print(type(10)) # int

print(type(3.14)) # float

print(type(1 + 1j)) # complex

print(type(True)) # bool

print(type([1, 2, 3, 4])) # list

print(type({'name':'Asabeneh','age':250, 'is\_married':250})) # dict

print(type((1,2))) # tuple

print(type(zip([1,2],[3,4]))) # set

**Casting**: Converting one data type to another data type. We use int(), float(), str(), list, set When we do arithmetic operations string numbers should be first converted to int or float otherwise it will return an error. If we concatenate a number with a string, the number should be first converted to a string. We will talk about concatenation in String section.

# int to float

num\_int = 10

print('num\_int',num\_int) # 10

num\_float = float(num\_int)

print('num\_float:', num\_float) # 10.0

# float to int

gravity = 9.81

print(int(gravity)) # 9

# int to str

num\_int = 10

print(num\_int) # 10

num\_str = str(num\_int)

print(num\_str) # '10'

# str to int or float

num\_str = '10.6'

print('num\_int', int(num\_str)) # 10

print('num\_float', float(num\_str)) # 10.6

# str to list

first\_name = 'Asabeneh'

print(first\_name) # 'Asabeneh'

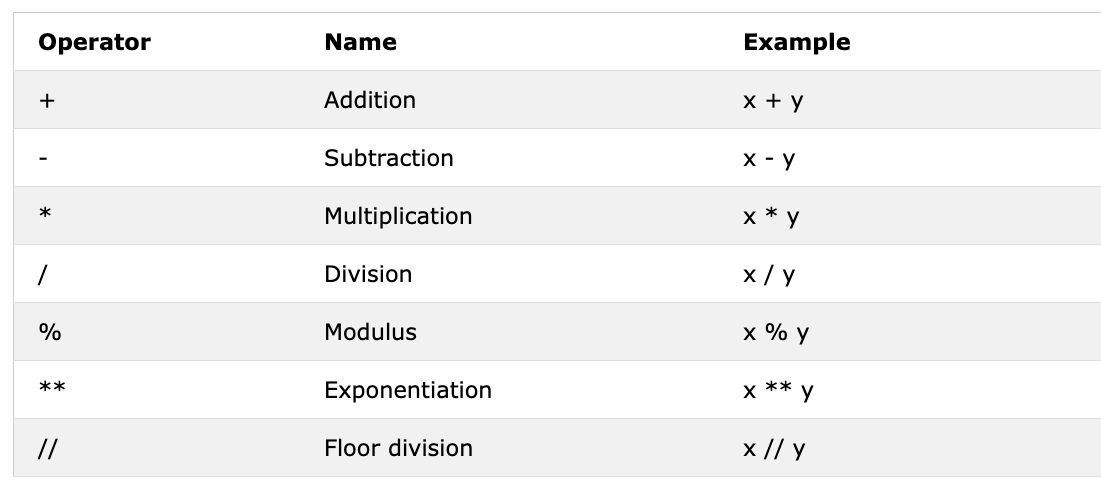
first\_name\_to\_list = list(first\_name)

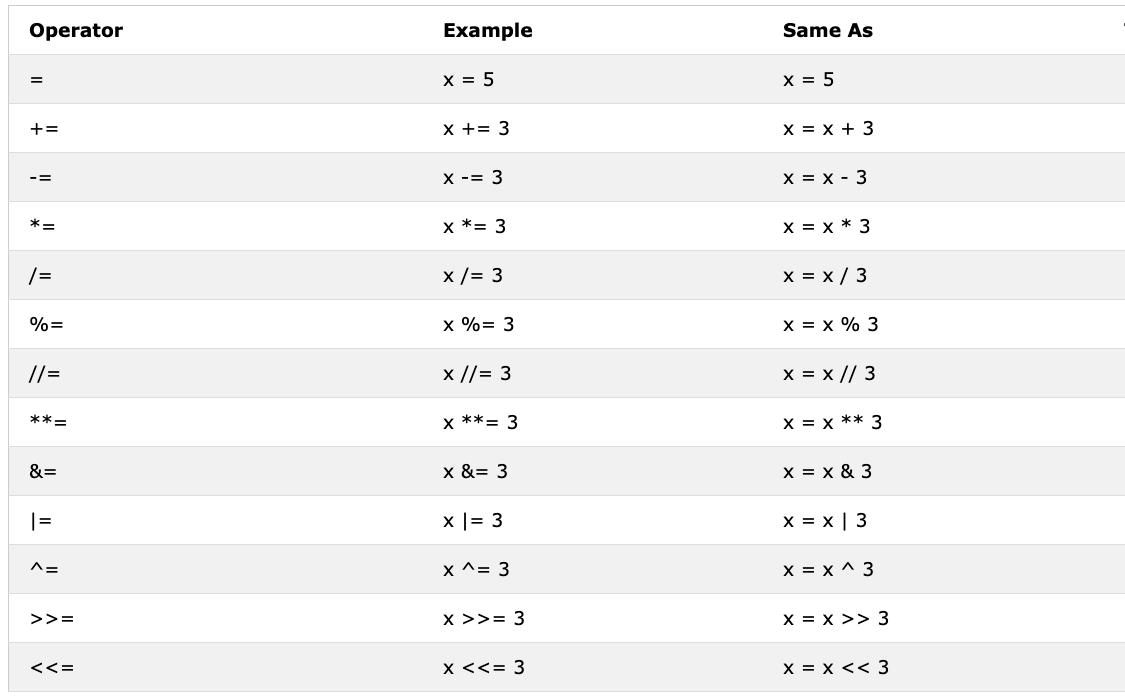
print(first\_name\_to\_list) # ['A', 's', 'a', 'b', 'e', 'n', 'e', 'h']

## Operators

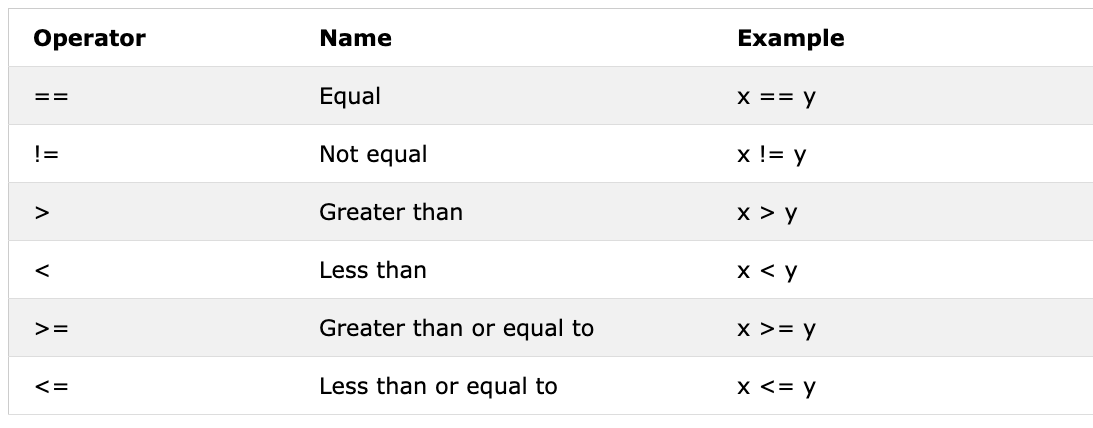
### Arithmetic Operators:

* Addition(+): a + b
* Subtraction(-): a - b
* Multiplication(\*): a \* b
* Division(/): a / b
* Modulus(%): a % b
* Floor division(//): a // b
* Exponentiation(\*\*): a \*\* b





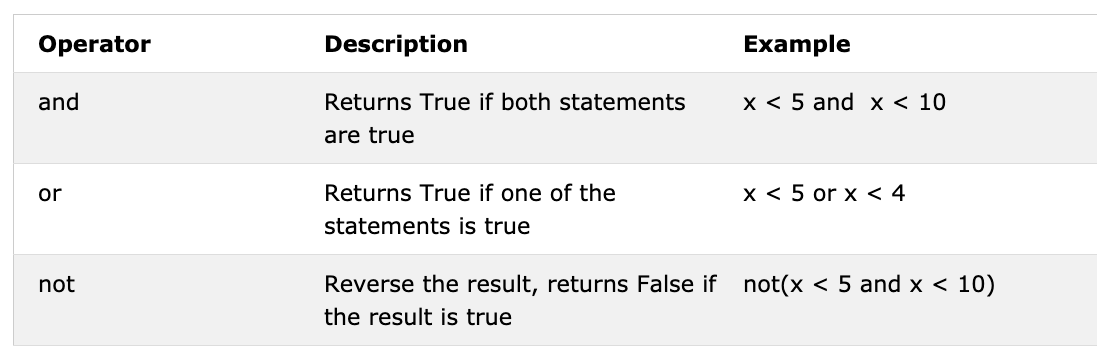
### Comparison Operators



**In addition to the above comparison operator Python uses:**

* *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)
* *in*: Returns True if the queried list contains a certain item(x in y)
* *not in*: Returns True if the queried list doesn't have a certain item(x in y)

### **Logical Operators**



## **Strings**

greeting = 'Hello, World!' # String could be made using a single or double quote,"Hello, World!"

print(greeting) # Hello, World!

print(len(greeting)) # 13

### String Concatenation

first\_name = 'Asabeneh'

last\_name = 'Yetayeh'

space = ' '

full\_name = first\_name + space + last\_name

print(full\_name) # Asabeneh Yetayeh

print(len(first\_name) > len(last\_name)) # True

### Escape Sequences in Strings

In Python and other programming languages \ followed by a character is an escape sequence. Let us see the most common escape characters:

* \n: new line
* \t: Tab means(8 spaces)
* \\: Back slash
* \': Single quote (')
* \": Double quote (")

#### New Style String Formatting (str.format)

This formatting is introduced in Python version 3.

first\_name = 'Asabeneh'

last\_name = 'Yetayeh'

language = 'Python'

formated\_string = 'I am {} {}. I teach {}'.format(first\_name, last\_name, language)

print(formated\_string)

### Python Strings as Sequences of Characters

#### Unpacking Characters

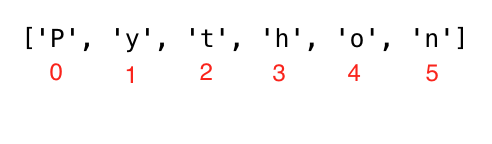
language = 'Python'

a,b,c,d,e,f = language # unpacking sequence characters into variables

print(a) # P

#### Accessing Characters in Strings by Index

In programming counting starts from zero. Therefore the first letter of a string is at zero index and the last letter of a string is the length of a string minus one.

[](https://github.com/Asabeneh/30-Days-Of-Python/blob/master/images/string_index.png)

language = 'Python'

first\_letter = language[0]

#### Slicing Python Strings

language = 'Python'

first\_three = language[0:3] # starts at zero index and up to 3 but not include 3

print(first\_three) #Pyt

#### Reversing a String

greeting = 'Hello, World!'

print(greeting[::-1]) # !dlroW ,olleH

#### Skipping Characters While Slicing

language = 'Python'

pto = language[0:6:2] #

print(pto) # Pto

### String Methods

* capitalize(): Converts the first character of the string to capital letter

challenge = 'thirty days of python'

print(challenge.capitalize()) # 'Thirty days of python'

* count(): returns occurrences of substring in string, count(substring, start=.., end=..). The start is a starting indexing for counting and end is the last index to count.

challenge = 'thirty days of python'

print(challenge.count('y')) # 3

* endswith(): Checks if a string ends with a specified ending

challenge = 'thirty days of python'

print(challenge.endswith('on')) # True

* expandtabs(): Replaces tab character with spaces, default tab size is 8. It takes tab size argument

challenge = 'thirty\tdays\tof\tpython'

print(challenge.expandtabs()) # 'thirty days of python'

print(challenge.expandtabs(10)) # 'thirty days of python'

* find(): Returns the index of the first occurrence of a substring, if not found returns -1

challenge = 'thirty days of python'

print(challenge.find('y')) # 16

* isalnum(): Checks alphanumeric character

challenge = 'ThirtyDaysPython'

print(challenge.isalnum()) # True

challenge = 'thirty days of python'

print(challenge.isalnum()) # False, space is not an alphanumeric character

* isalpha(): Checks if all string elements are alphabet characters (a-z and A-Z)

challenge = 'thirty days of python'

print(challenge.isalpha()) # False, space is once again excluded

challenge = 'ThirtyDaysPython'

print(challenge.isalpha()) # True

* isdecimal(): Checks if all characters in a string are decimal (0-9)

challenge = 'thirty days of python'

print(challenge.isdecimal()) # False

challenge = '123'

print(challenge.isdecimal()) # True

* islower(): Checks if all alphabet characters in the string are lowercase

challenge = 'thirty days of python'

print(challenge.islower()) # True

challenge = 'Thirty days of python'

print(challenge.islower()) # False

* isupper(): Checks if all alphabet characters in the string are uppercase

challenge = 'thirty days of python'

print(challenge.isupper()) # False

challenge = 'THIRTY DAYS OF PYTHON'

print(challenge.isupper()) # True

* join(): Returns a concatenated string

web\_tech = ['HTML', 'CSS', 'JavaScript', 'React']

result = ' '.join(web\_tech)

print(result) # 'HTML CSS JavaScript React'

* strip(): Removes all given characters starting from the beginning and end of the string

challenge = 'thirty days of pythoonnn'

print(challenge.strip('noth')) # 'irty days of py'

* replace(): Replaces substring with a given string

challenge = 'thirty days of python'

print(challenge.replace('python', 'coding')) # 'thirty days of coding'

* split(): Splits the string, using given string or space as a separator

challenge = 'thirty days of python'

print(challenge.split()) # ['thirty', 'days', 'of', 'python']

challenge = 'thirty, days, of, python'

print(challenge.split(', ')) # ['thirty', 'days', 'of', 'python']

* title(): Returns a title cased string

challenge = 'thirty days of python'

print(challenge.title()) # Thirty Days Of Python

* swapcase(): Converts all uppercase characters to lowercase and all lowercase characters to uppercase characters

challenge = 'thirty days of python'

print(challenge.swapcase()) # THIRTY DAYS OF PYTHON

challenge = 'Thirty Days Of Python'

print(challenge.swapcase()) # tHIRTY dAYS oF pYTHON

* startswith(): Checks if String Starts with the Specified String

challenge = 'thirty days of python'

print(challenge.startswith('thirty')) # True

challenge = '30 days of python'

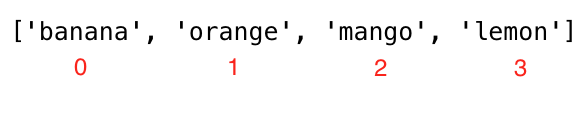
print(challenge.startswith('thirty')) # False

## **Lists**

lst = list()

lst = []

lst = ['Asabeneh', 250, True, {'country':'Finland', 'city':'Helsinki'}] # list containing different data types



fruits = ['banana', 'orange', 'mango', 'lemon']

first\_fruit = fruits[0] # we are accessing the first item using its index

print(first\_fruit) # banana

second\_fruit = fruits[1]

print(second\_fruit) # orange

lst = ['item','item2','item3', 'item4', 'item5']

first\_item, second\_item, third\_item, \*rest = lst

print(first\_item) # item1

print(second\_item) # item2

### Unpacking List Items

lst = ['item','item2','item3', 'item4', 'item5']

first\_item, second\_item, third\_item, \*rest = lst

print(first\_item) # item1

print(second\_item) # item2

### Slicing Items from a List

fruits = ['banana', 'orange', 'mango', 'lemon']

all\_fruits = fruits[0:4] # it returns all the fruits

# this will also give the same result as the one above

all\_fruits = fruits[0:] # if we don't set where to stop it takes all the rest

orange\_and\_mango = fruits[1:3] # it does not include the first index

orange\_mango\_lemon = fruits[1:]

orange\_and\_lemon = fruits[::2] # here we used a 3rd argument, step. It will take every 2cnd item - ['banana', 'mango']

### Modifying Lists

List is a mutable or modifiable ordered collection of items. Lets modify the fruit list.

fruits = ['banana', 'orange', 'mango', 'lemon']

fruits[0] = 'avocado'

print(fruits) # ['avocado', 'orange', 'mango', 'lemon']

fruits[1] = 'apple'

### Checking Items in a List

fruits = ['banana', 'orange', 'mango', 'lemon']

does\_exist = 'banana' in fruits

print(does\_exist) # True

### Adding Items to a List

lst = list()

lst.append(item)

### Inserting Items into a List

lst = ['item1', 'item2']

lst.insert(index, item)

fruits = ['banana', 'orange', 'mango', 'lemon']

fruits.insert(2, 'apple') # insert apple between orange and mango

### Removing Items from a List

lst = ['item1', 'item2']

lst.remove(item)

### Removing Items Using Pop

The pop() method removes the specified index, (or the last item if index is not specified):

lst = ['item1', 'item2']

lst.pop() # last item

lst.pop(index)

### Removing Items Using Del

The del keyword removes the specified index and it can also be used to delete items within index range. It can also delete the list completely

lst = ['item1', 'item2']

del lst[index] # only a single item

del lst # to delete the list completely

fruits = ['banana', 'orange', 'mango', 'lemon', 'kiwi', 'lime']

del fruits[0]

print(fruits) # ['orange', 'mango', 'lemon', 'kiwi', 'lime']

del fruits[1]

print(fruits) # ['orange', 'lemon', 'kiwi', 'lime']

del fruits[1:3] # this deletes items between given indexes, so it does not delete the item with index 3!

print(fruits) # ['orange', 'lime']

del fruits

print(fruits) # This should give: NameError: name 'fruits' is not defined

### Clearing List Items

The clear() method empties the list:

# syntax

lst = ['item1', 'item2']

lst.clear()

### Joining Lists

There are several ways to join, or concatenate, two or more lists in Python.

* Plus Operator (+)

# syntax

list3 = list1 + list2

positive\_numbers = [1, 2, 3, 4, 5]

zero = [0]

negative\_numbers = [-5,-4,-3,-2,-1]

integers = negative\_numbers + zero + positive\_numbers

print(integers) # [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]

* Joining using extend() method The *extend()* method allows to append list in a list. See the example below.
* num1 = [0, 1, 2, 3]
* num2= [4, 5, 6]
* num1.extend(num2)
* print('Numbers:', num1) # Numbers: [0, 1, 2, 3, 4, 5, 6]

negative\_numbers = [-5,-4,-3,-2,-1]

positive\_numbers = [1, 2, 3,4,5]

zero = [0]

negative\_numbers.extend(zero)

negative\_numbers.extend(positive\_numbers)

print('Integers:', negative\_numbers) # Integers: [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]