Python Complex Built-in Data Types

Shouke Wei, Ph.D. Professor

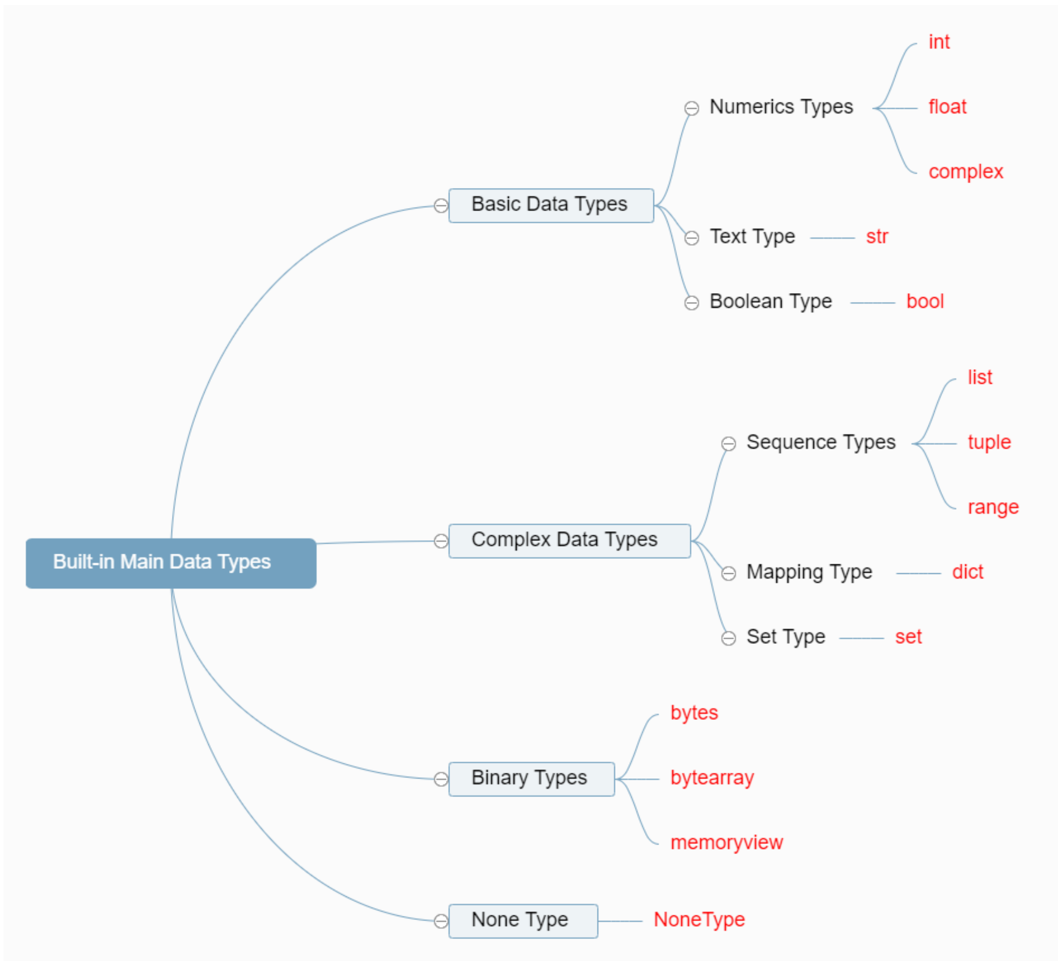
Email: shouke.wei@gmail.com

Objective

- This section will introduce Python Complex Built-in Data Types and some import methods.

1. Complex Built-in Data Types

- Multi Item Data Types, collection data types



2. Create Complex Data

2.1 List

- Lists are created using square brackets [ ]
- They can be any data types, String, integer, boolean and mixed ones
- A list items is ordered, changeable, and allow duplicate values

In [11]:
```
fruitList = ['Apple','Banana','Orange','Melon','Grape']
numberList = [1, 2, 3, 5, 0]
mixedList = [5, 'Apple', True, 6.0]

print(fruitList)
print(numberList)
print(mixedList)
```

```
['Apple', 'Banana', 'Orange', 'Melon', 'Grape']
[1, 2, 3, 5, 0]
[5, 'Apple', True, 6.0]
```

In [12]:
```
oneItemList = ['Apple']
print(oneItemList)
```

```
['Apple']
```

2.2 Tuple

- Tuples are written with round brackets ()
- They can be any data types, String, int, boolean and mixed ones
- Items are ordered, allow duplicate values, but unchangeable

In [13]:
```
fruitTuple = ('Apple','Banana','Orange','Melon','Grape')
numberTuple = (1, 3, 5, 7, 9)
booleanTuple = (True, False, False)
mixedTuple = ("Hello", 30, True, 80.5, "age")

print(fruitTuple)
print(numberTuple)
print(booleanTuple)
print(mixedTuple)
```

```
('Apple', 'Banana', 'Orange', 'Melon', 'Grape')
(1, 3, 5, 7, 9)
(True, False, False)
('Hello', 30, True, 80, 'age')
```

- Tuple with one item must have comma after the item

In [1]:
```
oneItemTuple = ('Apple',)
print(oneItemTuple)
```

```
('Apple',)
```

2.3 Range

- use `range ()` function to return a sequence of numbers
- range(start, stop, step)
  - start: optional, default is 0
  - stop: required
  - step: optional, default is 1

In [2]:
```
x = range(10)
print(x)
```

```
range(0, 10)
```

In [3]:
```
print(list(x))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [16]:
```
x = range(10)
for n in x:
  print(n)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [1]:
```
y = range(1,10,2)
print(list(y))
```

```
[1, 3, 5, 7, 9]
```

2.4 Dictionary

- Dictionaries are used to store data values in the pairs `key:value`
- written with curly brackets {}
- the values in dictionary items can be of any data type
- unordered, changeable and does not allow duplicates

In [18]:
```
fruitDict = {
 'type': 'Apple',
 'color': ['Red','Green','Yellow'],
 'sour':False,
 'sweet':True,
 'price':'2.0'}

print(fruitDict)
```

```
{'type': 'Apple', 'color': ['Red', 'Green', 'Yellow'], 'sour': False, 'sweet': True, 'price': '2.
0'}
```

2.5 Set

- Sets are written with curly brackets {}
- Set items can be of any data type
- Sets are unordered, unchangeable, and do not allow duplicate values

In [19]:
```
fruitSet = {'Apple','Banana','Orange','Melon','Grape'}
print(fruitSet)
```

```
{'Grape', 'Melon', 'Apple', 'Orange', 'Banana'}
```

3. Important Methods

3.1 Length Measure: `len()`

In [4]:
```python
fruitList = ['Apple','Banana','Orange','Melon','Grape']
fruitTuple = ('Apple','Banana','Orange','Melon','Grape')
x = range(10)
fruitDict = {
 'type': 'Apple',
 'color': ['Red','Green','Yellow'],
 'sour':False,
 'sweet':True,
 'price':'2.0'}
fruitSet = {'Apple','Banana','Orange','Melon','Grape'}
```

In [5]:
```python
print(len(fruitList))
print(len(fruitTuple))
print(len(x))
print(len(fruitDict))
print(len(fruitSet))
```

```
5
5
10
5
5
```

3.2 Type Check: `type()`

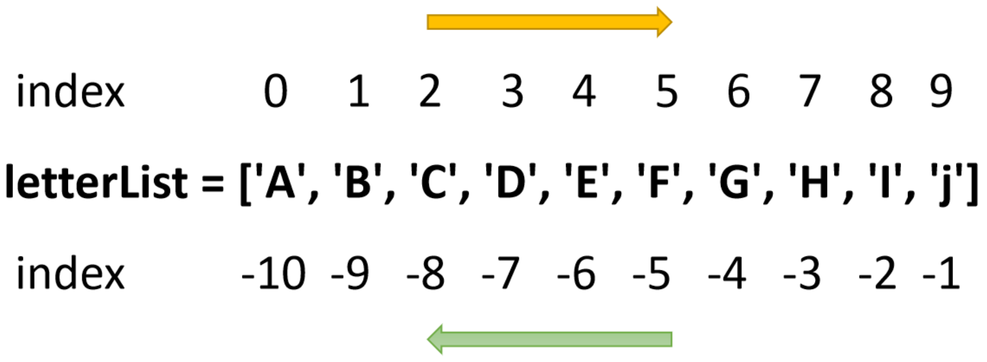
In [6]:
```python
fruit = ('Apple','Banana','Orange','Melon','Grape')
t = type(fruit)

print(t)
```

```
<class 'tuple'>
```

3.3 Access Items

(1) List and Tuple

- **items can be accessed by their index number**
- because List and Tuple items are indexed


index 0 1 2 3 4 5 6 7 8 9
letterList = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'j']
index -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

- The first item has index 0, -1 refers to the last item, -2 refers to the second last item etc
- a range of index [start:end] to access a range of a list or a tuple with start index included but end index excluded
- 0 index can be missed, write as [:end]

In [1]:
```python
nameList = ['Alice','Mike','Tome','John','Susan']
print(nameList[0])
print(nameList[-1])
print(nameList[2:4]) # index 2 included but index 4 not included
print(nameList[:4]) # from index 0
```

```
Alice
Susan
['Tome', 'John']
['Alice', 'Mike', 'Tome', 'John']
```

(2) set

- set values are unordered
- **items can be accessed by loop rather than index numbers**
- but loop through the set items using a for loop, or
- check if a specified value is in a set by using the `in` keyword

In [2]:
```
fruitSet = {'Apple','Banana','Melon','Orange','Melon','Grape'}

for item in fruitSet: # Loop through the set
    print(item)  # print all the items
```

```
Apple
Grape
Banana
Orange
Melon
```

(3) Dictionary

- **Items can be accessed by the key name** inside square brackets

In [4]:
```
fruitDic = {
 'type': 'Apple',
 'color': ['Red','Green','Yellow'],
 'sour':False,
 'sweet':True,
 'price':'2.0'}

print(fruitDic['price'])
```

```
2.0
```

- The `get()` method can get the same result

In [5]:
```
x = fruitDic.get('price')

print(x)
```

```
2.0
```

- The `keys()` method returns a list of all the keys in the dictionary

In [8]:
```
keys = fruitDic.keys()

print(keys)
```

```
dict_keys(['type', 'color', 'sour', 'sweet', 'price'])
```

- The `values()` method returns a list of all the values in the dictionary

In [9]:
```
values = fruitDic.values()
print(values)
```

```
dict_values(['Apple', ['Red', 'Green', 'Yellow'], False, True, '2.0'])
```

3.4 Change items

(1) List

- change items also using their index

In [1]:
```
fruitList = ['Apple','Banana','Orange','Melon','Grape']

fruitList[0] = 'Cherry'
print(fruitList)
```

```
['Cherry', 'Banana', 'Orange', 'Melon', 'Grape']
```

- Change multiple items

In [4]:
```
fruitList = ['Apple','Banana','Orange','Melon','Grape']
fruitList[0:1] = ['Cherry','Watermelon']
print(fruitList)
```

```
['Cherry', 'Watermelon', 'Banana', 'Orange', 'Melon', 'Grape']
```

In [5]:
```
fruitList = ['Apple','Banana','Orange','Melon','Grape']
fruitList[0:2] = ['Cherry','Watermelon']
print(fruitList)
```

```
['Cherry', 'Watermelon', 'Orange', 'Melon', 'Grape']
```

(2) Tuple

- Tuples are unchangeable, or immutable. so
- convert the tuple into a list, change the list, and convert the list back into a tuple

In [6]:
```
fruitTuple = ('Apple','Banana','Orange','Melon','Grape')
fruitList = list(fruitTuple)
fruitList[1] = "Cherry"
fruitTuple = tuple(fruitList)

print(fruitTuple)
```

```
('Apple', 'fruitList', 'Orange', 'Melon', 'Grape')
```

(3) Set

- Once a set is created, you cannot change its items
- convert the set into a list, change it and convert it back into a set

In [32]:
```
fruitSet = {'Apple','Banana','Orange','Melon','Grape'}
fruitList = list(fruitSet)
print(fruitList)
```

```
['Grape', 'Banana', 'Orange', 'Apple', 'Melon']
```

In [33]:
```
fruitList[3] = "Cherry"
fruitSet = set(fruitList)

print(fruitSet)
```

```
{'Cherry', 'Orange', 'Melon', 'Grape', 'Apple'}
```

(4) Dictionary

- change the value of a specific item by referring to its key name

In [5]:
```
fruitDic = {
 'type': 'Apple',
 'color': ['Red','Green','Yellow'],
 'sour':False,
 'sweet':True,
 'price':'2.0'}
```

In [6]:
```
fruitDic['price'] = 3.0
print(fruitDic)
```
```
{'type': 'Apple', 'color': ['Red', 'Green', 'Yellow'], 'sour': False, 'sweet': True, 'price': 3.0}
```

- The `update()` method updates the dictionary with the items from the given argument

In [7]:
```
fruitDic = {
 'type': 'Apple',
 'color': ['Red','Green','Yellow'],
 'sour':False,
 'sweet':True,
 'price':2.0}
fruitDic.update({"price": 3.0})
print(fruitDic)
```
```
{'type': 'Apple', 'color': ['Red', 'Green', 'Yellow'], 'sour': False, 'sweet': True, 'price': 3.0}
```

3.5 Add items

(1) List

- the `append()` method adds an item to the end of the list
- the `insert()` method inserts a new list item at the specified index

In [36]:
```
# append Item
fruitList = ['Apple','Banana','Orange','Melon','Grape']
fruitList.append('Cherry')

print(fruitList)
```
```
['Apple', 'Banana', 'Orange', 'Melon', 'Grape', 'Cherry']
```

In [1]:
```
# Insert Items
fruitList = ['Apple','Banana','Orange','Melon','Grape']

fruitList.insert(1, 'Cherry')
print(fruitList)
```
```
['Apple', 'Cherry', 'Banana', 'Orange', 'Melon', 'Grape']
```

(2) Tuple

- The processes can be done by converting the tuple into a list, change the list, and convert the list back into a tuple, e.g.

In [18]:
```
# append an item
fruitTuple = ('Apple','Banana','Orange','Melon','Grape')

fruitList = list(fruitTuple)
fruitList.append('Cherry')
print(fruitList)

fruitTuple_new = tuple(fruitList)
print(fruitTuple_new)
```
```
['Apple', 'Banana', 'Orange', 'Melon', 'Grape', 'Cherry']
```

In [22]:
```
# insert an item
fruitTuple = ('Apple','Banana','Orange','Melon','Grape')

fruitList = list(fruitTuple)
fruitList.insert(2,'Watermelon')
print(fruitList)

fruitTuple2= tuple(fruitList)
print(fruitTuple2)
```

```
['Apple', 'Banana', 'Watermelon', 'Orange', 'Melon', 'Grape']
('Apple', 'Banana', 'Watermelon', 'Orange', 'Melon', 'Grape')
```

(3) Set

- Once a set is created, you cannot change its items, but
- you can add new items by using `add()`

In [10]:
```
fruitSet = {'Apple','Banana','Melon','Orange','Grape'}

fruitSet.add('Cherry')
print(fruitSet)
```

```
{'Cherry', 'Melon', 'Orange', 'Apple', 'Banana', 'Grape'}
```

(4) Dictionary

- Adding an item to the dictionary by using a new index key and a value

In [24]:
```
fruitDic = {
 'type': 'Apple',
 'color': ['Red','Green','Yellow'],
 'sour':False,
 'sweet':True,
 'price':'2.0'}
fruitDic['origin'] = 'USA'
print(fruitDic)
```

```
{'type': 'Apple', 'color': ['Red', 'Green', 'Yellow'], 'sour': False, 'sweet': True, 'price': '2.
0', 'origin': 'USA'}
```

- The `update()` method updates the dictionary with the items from the given argument

In [25]:
```
fruitDic.update({"In stock": 'Yes'})
print(fruitDic)
```

```
{'type': 'Apple', 'color': ['Red', 'Green', 'Yellow'], 'sour': False, 'sweet': True, 'price': '2.
0', 'origin': 'USA', 'In stock': 'Yes'}
```

- The `update()` method updates the dictionary with the items from a given argument
- If the item does not exist, the item will be added

3.6 Join or Merge

- one or more data types

(1) lists

In [29]:
```python
# Join two or more lists

fruitList = ['Apple','Banana','Orange','Melon','Grape']
tropFruits = ["mango", "pineapple", "papaya"]

fruitList_new = fruitList + tropFruits

print(fruitList_new)
```

```
['Apple', 'Banana', 'Orange', 'Melon', 'Grape', 'mango', 'pineapple', 'papaya']
```

In [31]:
```python
# extend method
fruitList = ['Apple','Banana','Orange','Melon','Grape']
tropFruits = ["mango", "pineapple", "papaya"]

fruitList.extend(tropFruits)
print(fruitList)
```

```
['Apple', 'Banana', 'Orange', 'Melon', 'Grape', 'mango', 'pineapple', 'papaya']
```

- The extend() method is not only to append lists, but also to add any iterable object, such as tuples, sets, dictionaries, etc.

In [32]:
```python
# append a tuple to a list
fruitList2 = ['Apple','Banana','Orange','Melon','Grape']
fruitTuple = ("mango", "pineapple", "papaya")

fruitList2.extend(fruitTuple)
print(fruitList2)
```

```
['Apple', 'Banana', 'Orange', 'Melon', 'Grape', 'mango', 'pineapple', 'papaya']
```

(2) Tuples

In [19]:
```python
# add another tuple with one element or more
fruitTuple1 = ('Apple','Banana','Orange','Melon','Grape')
fruitTuple2 = ("mango", "pineapple", "papaya")
fruitTuple = fruitTuple1 + fruitTuple2
print(fruitTuple)
```

```
('Apple', 'Banana', 'Orange', 'Melon', 'Grape', 'mango', 'pineapple', 'papaya')
```

In [39]:
```python
# extend method does not work for tuple
fruitTuple1 = ('Apple','Banana','Orange','Melon','Grape')
fruitTuple2 = ("mango", "pineapple", "papaya")

fruitTuple1.extend(fruitTuple)
print(fruitTuple1)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_9152/2416216915.py in <module>
      3 fruitTuple2 = ("mango", "pineapple", "papaya")
      4 
----> 5 fruitTuple1.extend(fruitTuple)
      6 print(fruitTuple1)

AttributeError: 'tuple' object has no attribute 'extend'
```

(3) Sets

- Join Sets or a set with any other iterable object (tuples, lists, dictionaries etc.) by using `update()`

In [40]:
```python
# join sets
fruitSet1 = {'Apple','Banana','Melon'}
fruitSet2 = {'Orange','Melon','Grape'}

fruitSet1.update(fruitSet2)
print(fruitSet1)
```

```
{'Melon', 'Orange', 'Apple', 'Banana', 'Grape'}
```

In [37]:
```python
# merge a list into the current set
fruitSet = {'Apple','Banana','Melon'}

fruitList = ['Orange','Melon','Grape']

fruitSet.update(fruitList)
print(fruitSet)
```

```
{'Orange', 'Banana', 'Grape', 'Melon', 'Apple'}
```

(4) Dictionaries

- The `update()` method merge two or more dictionaries

In [20]:
```python
dict_1 = {'John': 15, 'Rick': 10, 'Misa' : 12 }
dict_2 = {'Bonnie': 18,'Rick': 20,'Matt' : 16 }
dict_1.update(dict_2)
print('Merge two dictionaries:')
print(dict_1)
```

```
Merge two dictionaries:
{'John': 15, 'Rick': 20, 'Misa': 12, 'Bonnie': 18, 'Matt': 16}
```