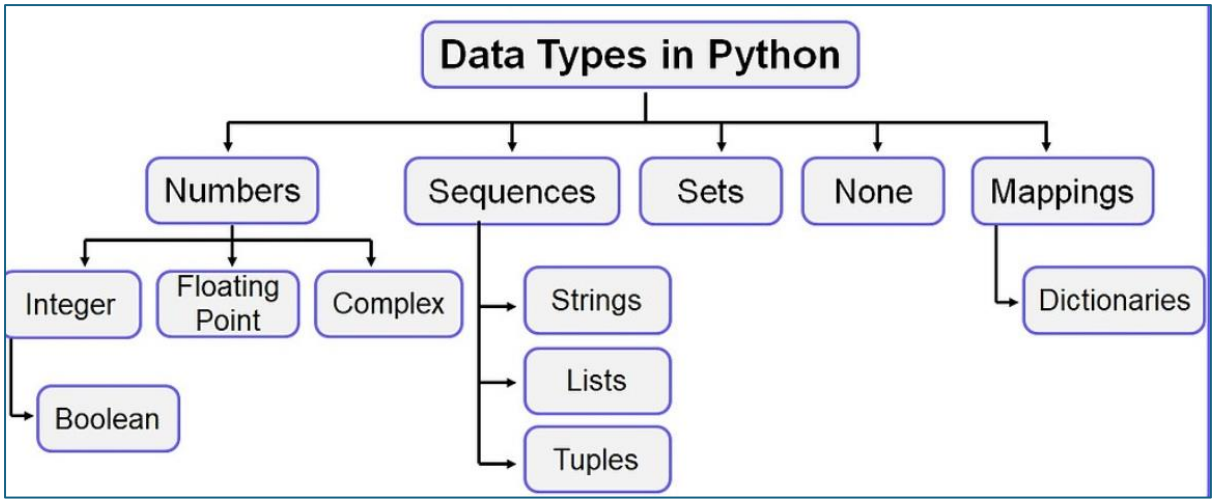
Datatypes:



Code to understand:

```
#code snippet to show the datatypes in python

a = 10                                                                  # int
b = 23.4566                                                             # float
c = "Hello Multimise"                                                   # string
d = (10 + 0j)                                                           # complex
e = [1, 2, 3, 4, 5, 6, 7, "Grow", "Learn", "Explore", 1114.67, (19+0j)] # List
f = (1, 2, 3, 4, 5, 6, 7, "Grow", "Learn", "Explore", 1114.67, (19+0j)) # Tuple
g = {0, 1, 2, 3, 4}                                                     # set
h = calories = {'apple' : 52, 'banana' : 89, 'choco' : 546}             # Dictionary
k = True                                                                # Boolean

allInOne = [a, b, c, d, e, f, g, h, k]

for i in allInOne[:]:
    print('Object Name = ', i, ' and it belongs to ',type(i), sep=' ')
    print('-' * 100)
```

Lists – Basics and complete walkthrough – Understand through code

```
from copy import copy
from copy import deepcopy

# List - Complete walkthrough
# creating the List
x = 0
myList = [1, 2, 3, 4, 5, 10, "green"]       # using LIST literal
numbers = list(range(100))                  # using list() constructor

#accessing the elements
print('myList[0] = myList[-7] = ', myList[-7])          # elements can be accessed from index 0 to Length (positive - normal)
print('myList[6] = myList[-1] = ', myList[-1])          # elements can also be accessed from -1 to -length (negative - backwards)
print(numbers[99])                                      # In the range of 100 numbers, the 99th index position contains '100'

# traversing the elements
for j in myList:
    print(j)

# traversing the range of 100 numbers and counting the multiples of 5
for k in numbers:
    if k % 5 == 0:
        x += 1
print(x, 'number of multiples of 5 is in the range of 100 numbers')

# updating the elements in the list

myList[2] = "Heavens"

print(myList)

# creating alias of the list

a = [5, 10, 15, 20, 25]
newLst = a

print(newLst)

#update the new list and try printing the old one

newLst[3] = "Good day"

print(newLst)

# creating shallow copy of the list

countries = ["United States", "Canada", "Poland", "Germany", "Austria"]

nations = countries.copy()    # nations = countries[:]

print("Address of nations: ", id(nations))
print("Address of countries: ", id(countries))

print("\n","Printing the address of countries")
for h in countries[:]:
    print(h, id(h))

print("\n","Printing the address of nations")
for t in nations[:]:
    print(t, id(t))

print((id(countries) == id(nations)))
print((id(nations[0]) == id(countries[0])))
print((id(nations[1]) == id(countries[1])))
```

List – Methods:

```
# Working with the List methods

# create a new list
animals = ['dog', 'cat']

# append() : adds an element to the end of the list.                                    .append(item)
animals.append('cow')
animals.append('Bull')                                  # animals[len(animals):] = ["hawk"]

print(animals)

# extend() : extends the list by appending elements from another iterable.              .extend([iterables])
animals.extend(['Goat', 'pigs', 'dragons'])             # animals[len(animals):] = ['Goat', 'pigs', 'dragons']

print(animals)

# insert() : adds an element at a specified position in the list.                       .insert(index, item)
animals.insert(5, 'Hens')                               # list_object[index:index] = [item]

# remove() : removes the first occurrence of a specified element from the list.         .remove(item)
animals.remove("pigs")

print(animals)

# pop()    : removes the element at a specified position in the list and returns it.    .pop([index])
animals.pop()
print(animals)

animals.pop(0)
animals.pop(-1)

print(animals)

# sort()   : sorts the elements in the list in ascending order.


# reverse(): reverses the order of the elements in the list.
# It's important to note that reversed() retrieves items from the input sequence lazily. This means that if something changes in the input sequence during the reversing # # process, then those changes will reflect in the final result.

g = reveresed(animals)
print(type(g))
print(list(g))

# count()  : returns the number of times a specified element appears in the list.
cnt = animals.count()
print("count of animals in farm: ", cnt)

animals.append("cows")
cnt_cows = animals.count("cows")
print("count of cows in farm: ", cnt_cows)

# index()  : returns the index of the first occurrence of a specified element in the list.
pos = animals.index("Bull")
print(pos)

# clear : Remove all the items in one go.                                                .clear()
animals.clear()                                         # animals[:] = []
```

List – Nested tuple and dict:

```
# Nested sequence (tuple) and dictionary in the list

employees = [
     ("Sanmaya", 20, "Software Engineer"),
     ("Aadhi", 25, "Web Developer"),
     ("Gayathri", 23, "Data Analyst"),
     ("Mark Antony", 22, "Intern"),
     ("Subramani", 30, "Project Manager")
 ]

#To access Gayathri's record - Get Gayathri's role
print('Gayathri', "'s", 'role in Multimise: ', employees[2][2])

# To access subramani's record - Use negative indexing
print('Subramani',"'s", 'role in Multimise: ', employees[-1][-1])

#Traversing the nested sequence in the list
for name, age, role in employees:
    print('Name = ',name, ', Age = ', age, ', Role = ', role)

# Dict inside the List
employees = [
     {"name": "Sanmaya", "age": 30, "job": "Software Engineer"},
     {"name": "Aadhi", "age": 25, "job": "Web Developer"},
     {"name": "Gayathri", "age": 45, "job": "Data Analyst"},
     {"name": "Mark Antony", "age": 22, "job": "Intern"},
     {"name": "Subramani", "age": 36, "job": "Project Manager"}
]

# Try to access Mark's record

print(employees[3]["name"])
print(employees[3]["age"])
print(employees[3]["job"])
```

List – Slicing

```
# Slicing of the list

# Syntax -> list_object[start:stop:step]

#create a list

a = list((1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

print(a)

print(a[-4])
print(a[4])
print(a[-4:])
print(a[-4:-2])

# Just print the list
print(a)

# Print only even numbers
print(a[1::2])

# print only odd numbers
print(a[0::2])

# as the starting position doesn't change, just step up to 2 to get odd numbers
print(a[::2])

# print the list using sliceing
print(a[:])
print(a[::])

# get the middle four
print(a[3:7])

# get the first three even numbers
print(a[1:6:2])

# get the first three even numbers - use negative indexing
print(a[-9:6:2])

# get the first three odd numbers - use negative indexing
print(a[:6:2])

# If start is before the beginning of the list, which can happen when you use negative indices, then Python will use 0 instead.
print(a[-11:2])
print(a[-11:1])
print(a[-100:2])

# If start is greater than stop, then the slicing will return an empty list.
print(a[1:-10])

# If stop is beyond the length of the list, then Python will use the length of the list instead.
print(a[:11])

# print the list in reverse way - negative value to step counter
print(a[::-1])
print(a[::-2])
print(a[::-3])
print(a[::-4])
print(a[1::-4])
```