# Hi! I am Python

# Python Programming Language

Hi I am Python
Programming Language

## Table Of Contents

## Variables in Memory

a = 10

memory for 10 is 12345

b = 10

memory for 10 is 12345

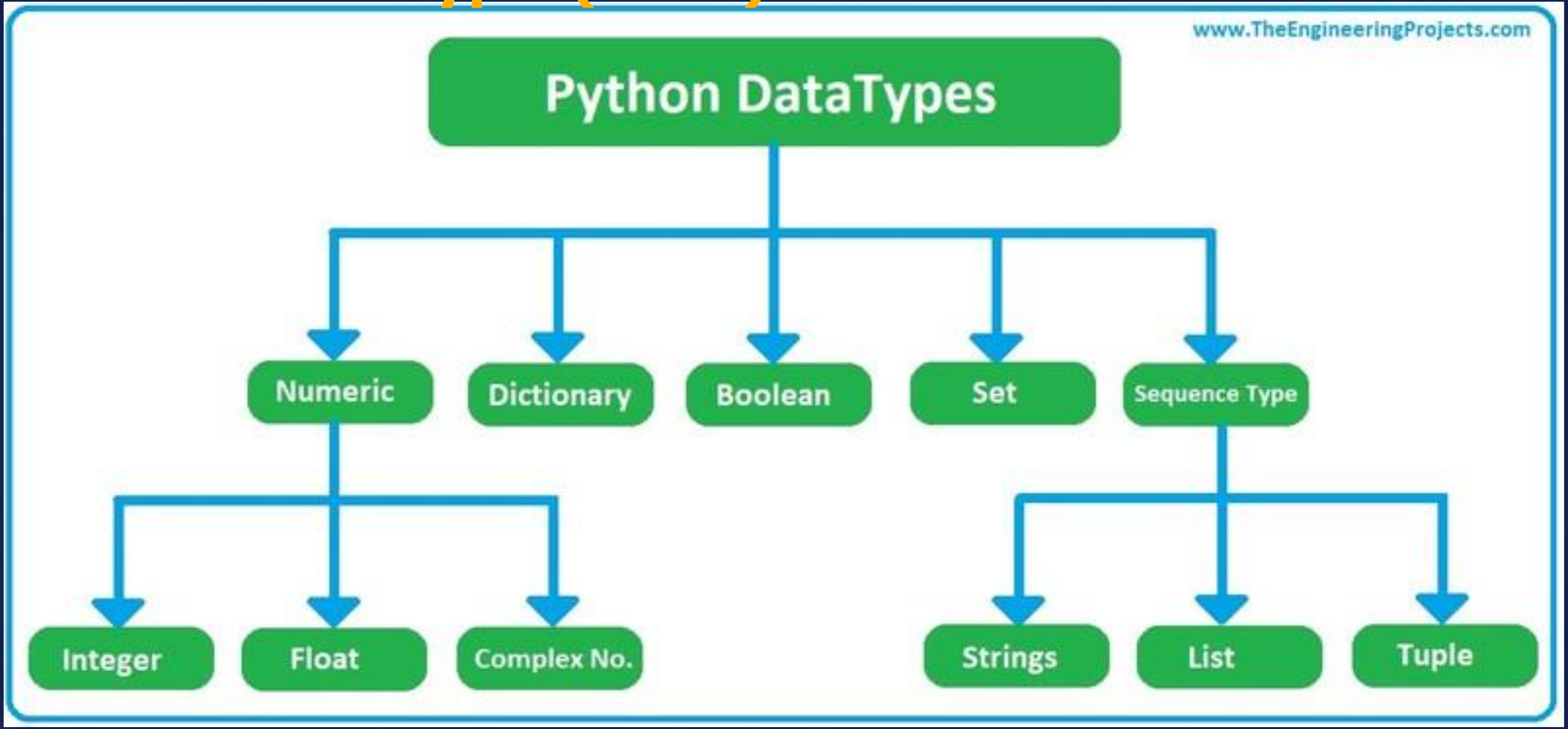
a = 12

memory for 12 is 12347

b = 11

memory for 11 is 12346

## Built in Datta Types(cont.)



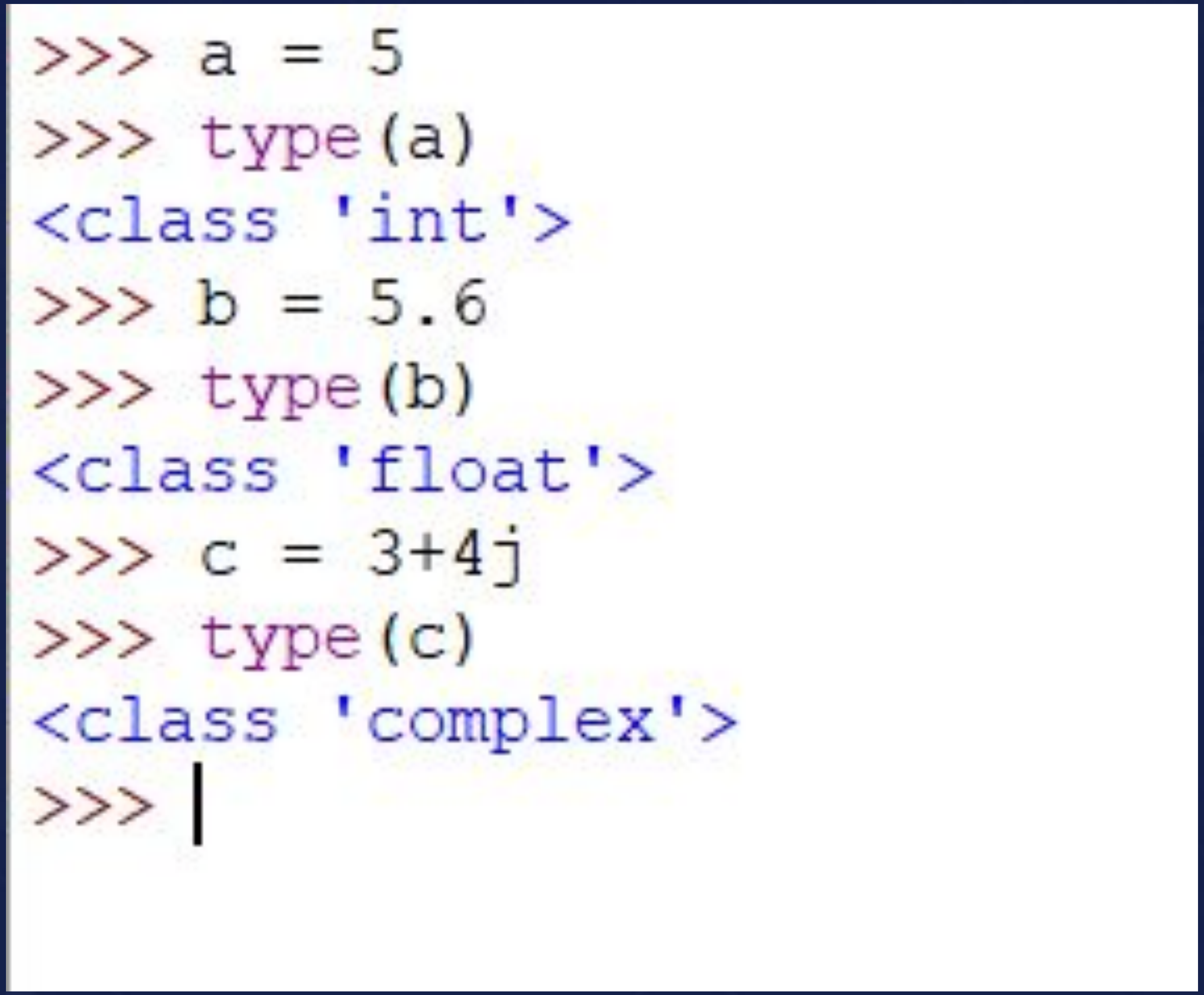
## Numeric Data Type

numeric data type represent the data which has numeric value. Numeric value can be integer, floating number or even complex numbers. These values are defined as int, float and complex class in Python.

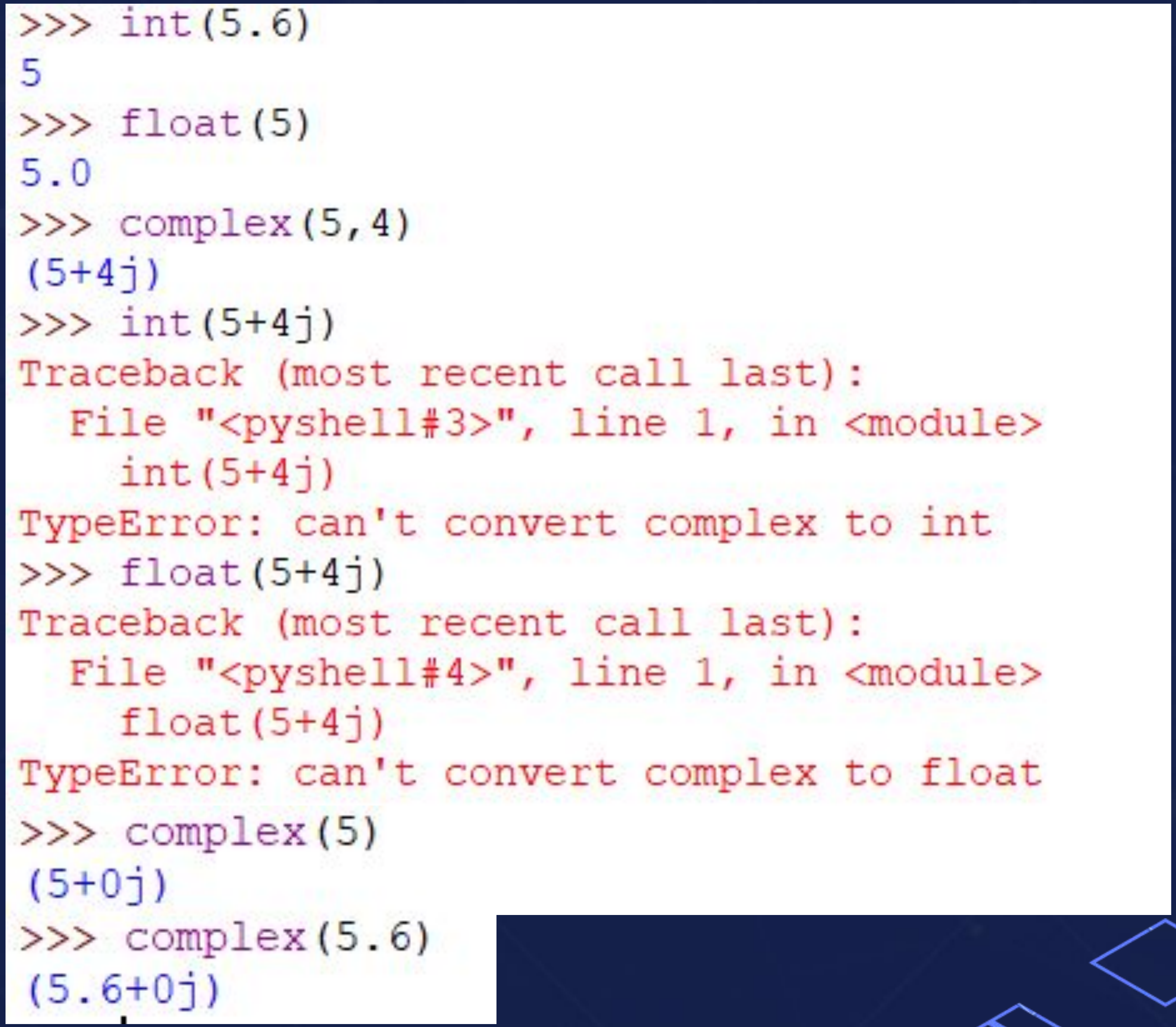
- **Integers** – This value is represented by int class. **In Python there is no limit to how long an integer value can be**. **-3,-2,-1,0,1,2,3 . int()**
- **Float** – This value is represented by float class. It is a real number with floating point representation. **9.34,9.00,3.45 . float()**
- **Complex Numbers** – Complex number is represented by complex class. It is specified as (real part) + (imaginary part)j. For example – **2+3j . complex()**

## Numeric Data type example using type() function:

type checking : type conversion :



```
>>> a = 5
>>> type(a)
<class 'int'>
>>> b = 5.6
>>> type(b)
<class 'float'>
>>> c = 3+4j
>>> type(c)
<class 'complex'>
>>>
```



```
>>> int(5.6)
5
>>> float(5)
5.0
>>> complex(5,4)
(5+4j)
>>> int(5+4j)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    int(5+4j)
TypeError: can't convert complex to int
>>> float(5+4j)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    float(5+4j)
TypeError: can't convert complex to float
>>> complex(5)
(5+0j)
>>> complex(5.6)
(5.6+0j)
```

## Sequence Type

In Python, sequence is the **ordered collection of similar or different data types**.

1) **String : (immutable)**
A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by **str** class. example : **'hello', "hello", """hello""". str()**

2) **List : (mutable)**
Lists are just like the arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type . **example : data = [1,2,3,4,5,'food','fruits', 4,5,'food']. list()**

## Creating List

**# Creating a List**

List = []

**#creating list using list() method/function:**

List = list("hello") → ['h', 'e', 'l', 'l', 'o']

**# Creating a List with the use of a String**

List = ['workingwithlist']

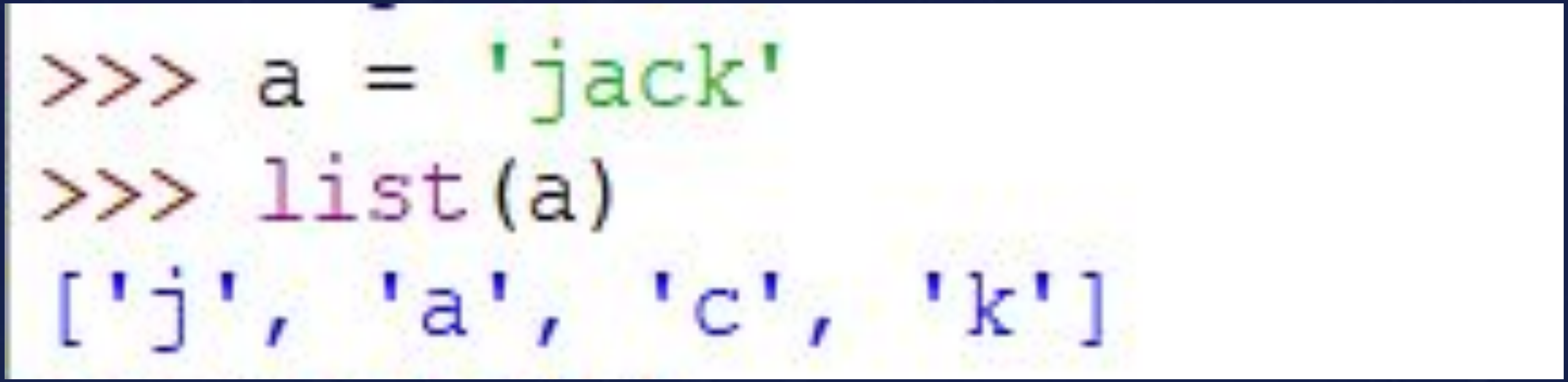
**# Creating a List with the use of multiple values**

List = ["working", "with", "list"]

**# Creating a Multi-Dimensional List (By Nesting a list inside a List)**

List = [['working', 'with'], ['list']]

## Type Conversion



```
>>> a = 'jack'
>>> list(a)
['j', 'a', 'c', 'k']
```

## Creating Tuple

**3) Tuple : (immutable)**
Just like list, tuple is also an ordered collection of Python objects. The only difference between tuple and list is that tuples are **immutable** i.e. **tuples cannot be modified after it is created.** It is represented by tuple class. tuple()
**example : (1,2,3,4,'food', 4,5)**

## Creating Tuple

**# Creating an empty tuple**

Tuple1 = ()

**# Creating a Tuple with the use of Strings**

Tuple1 = ('coding', 'For')

**# Creating a Tuple with the use of list**

list1 = [1, 2, 4, 5, 6]
tup1 =
tuple([1,2,3,4,5,6]) → (1, 2, 4, 5, 6)

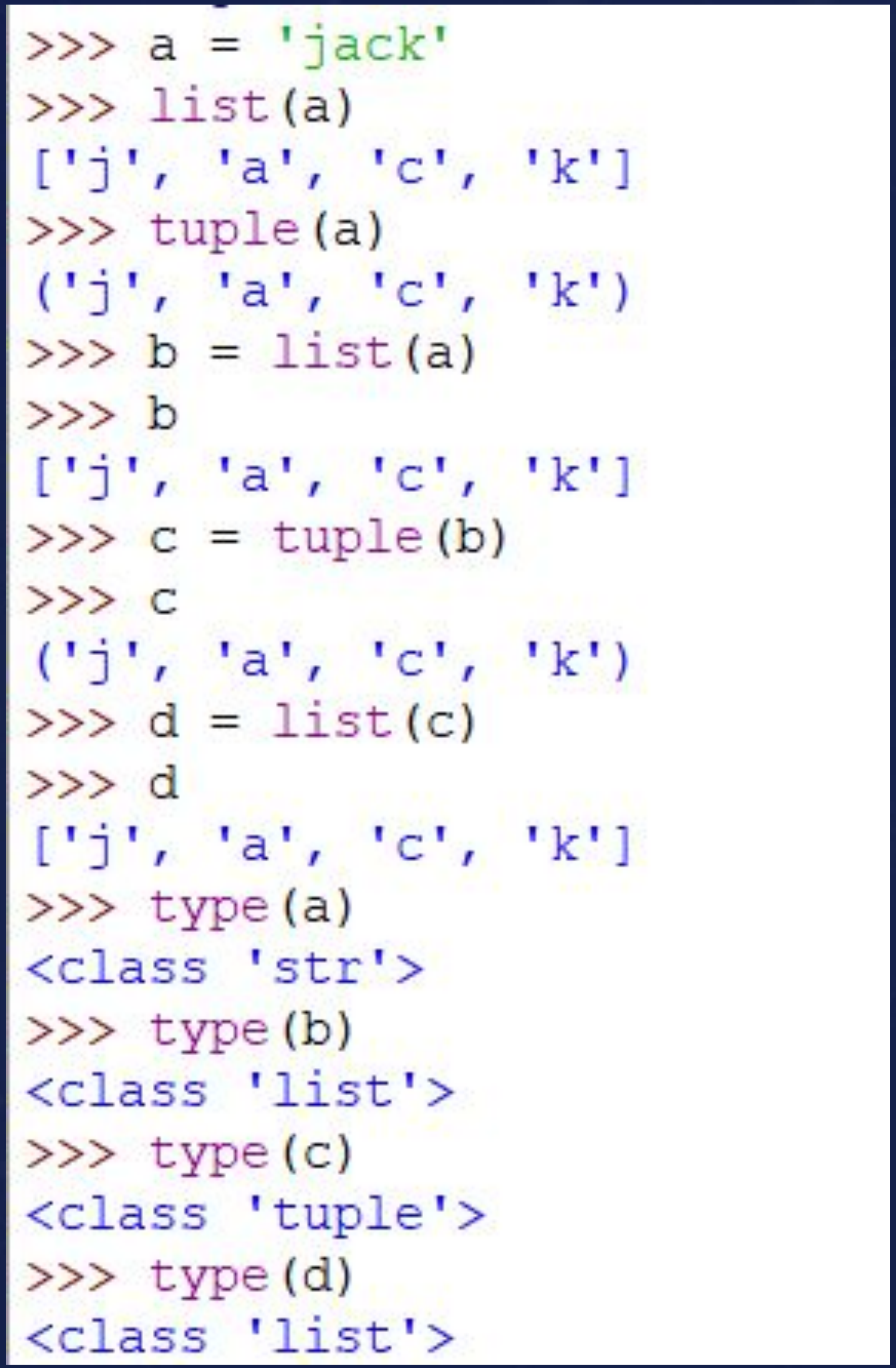
**# Creating a Tuple with the use of built-in function**

Tuple1 = tuple('python') → ('p', 'y', 't', 'h', 'o', 'n')

**# Creating a Tuple with nested tuples**

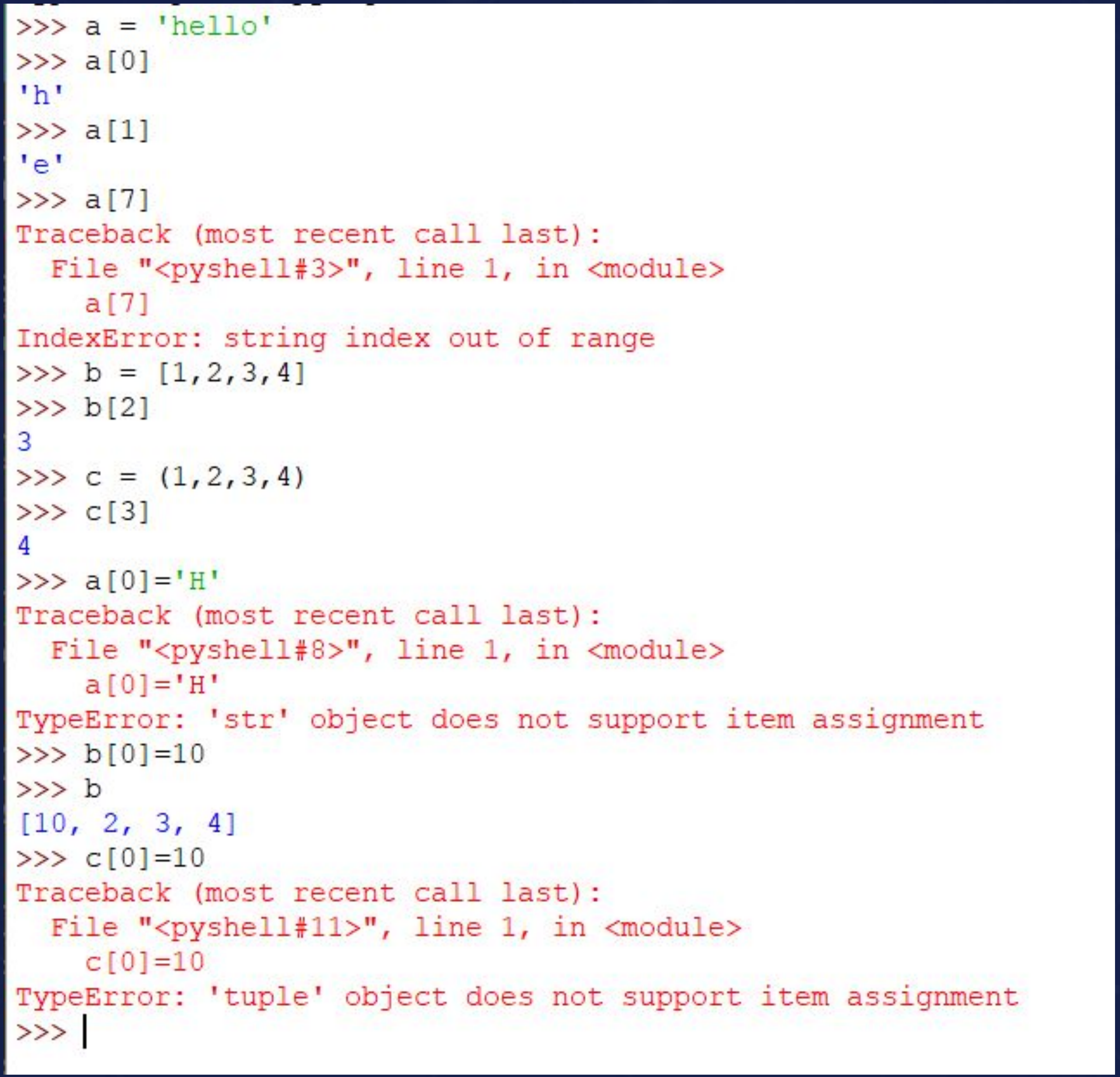
Tuple1 = (0, 1, 2, 3)
Tuple2 = ('python', 'geek')
Tuple3 = (Tuple1, Tuple2)

# Type Conversion



```
>>> a = 'jack'
>>> list(a)
['j', 'a', 'c', 'k']
>>> tuple(a)
('j', 'a', 'c', 'k')
>>> b = list(a)
>>> b
['j', 'a', 'c', 'k']
>>> c = tuple(b)
>>> c
('j', 'a', 'c', 'k')
>>> d = list(c)
>>> d
['j', 'a', 'c', 'k']
>>> type(a)
<class 'str'>
>>> type(b)
<class 'list'>
>>> type(c)
<class 'tuple'>
>>> type(d)
<class 'list'>
```

# indexing



```
>>> a = 'hello'
>>> a[0]
'h'
>>> a[1]
'e'
>>> a[7]
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    a[7]
IndexError: string index out of range
>>> b = [1,2,3,4]
>>> b[2]
3
>>> c = (1,2,3,4)
>>> c[3]
4
>>> a[0]='H'
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    a[0]='H'
TypeError: 'str' object does not support item assignment
>>> b[0]=10
>>> b
[10, 2, 3, 4]
>>> c[0]=10
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    c[0]=10
TypeError: 'tuple' object does not support item assignment
>>> 
```

## Set

Set is an **unordered** and **unindexed** collection of items in Python. **iterable, mutable and has no duplicate elements**. Unordered means when we display the elements of a set, it will come out in a random order. Unindexed means, we cannot access the elements of a set using the indexes like we can do in list and tuples.

example : {1,2,3,4}

## Set

```
# Creating a Set
set1 = set()
print("Initial blank Set: ", set1)
```

→

```
Initial blank Set: set()
```

```
# Creating a Set with the use of a String
set1 = set("phiiittrrrooonn")
```

→

```
{'p', 'h', 't', 'i', 'r', 'o',
'n'}
```

```
# Creating a Set with the use of a List
set1 = set(["learn", "and", "learn"])
```

→

```
{'learn', 'and'}
```

```
# Creating a Set with a mixed type of values(Having numbers and strings)
set1 = set([1, 2, 'hello', 4, 'world', 6, 'hello', 'Hello'])
```

→

```
{'hello', 1, 2, 4, 6, 'world', 'Hello'}
```

## Dictionary

Dictionary in Python is an **unordered collection** of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a ‘comma’.

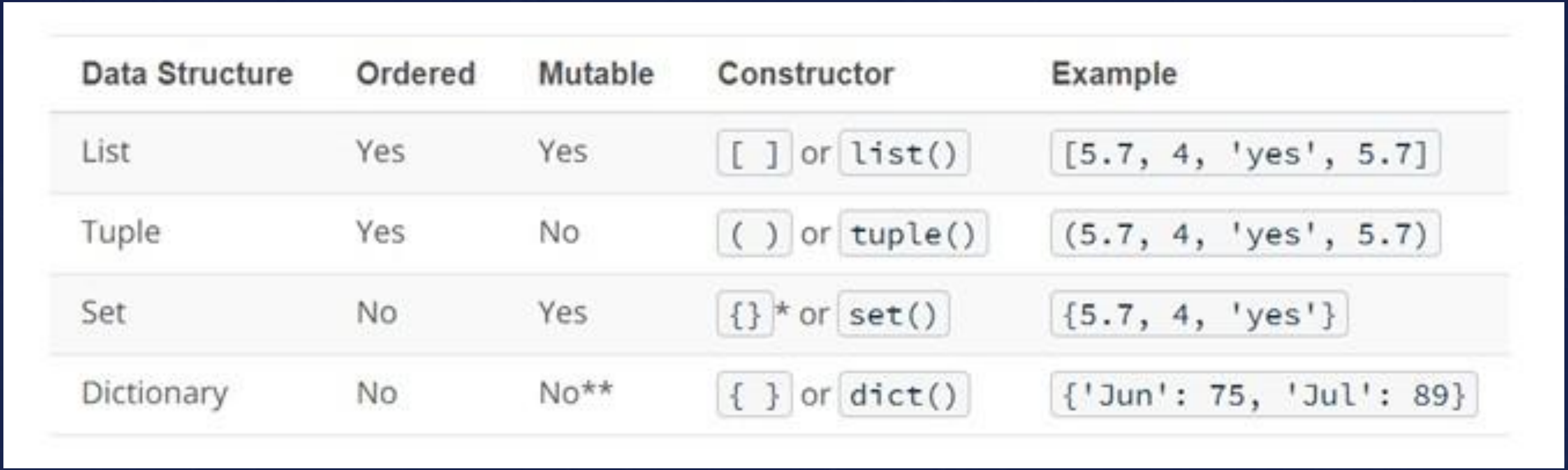
**{1: 'Learn', 2: 'with', 3: 'encodemy'}**

- **no duplicate key is allowed**
- **The values in the dictionary can be of any type, while the keys must be immutable like numbers, tuples, or strings.**
- **Dictionary keys are case sensitive- Same key name but with the different cases are treated as different keys in Python dictionaries.**

## Creating Dictionary

```
# Creating an empty Dictionary
Dict = {}

# Creating a Dictionary with Integer Keys
Dict = {1: 'Learn', 2: 'with', 3: 'phitron'}

# Creating a Dictionary with Mixed keys
Dict = {'Name': 'phitron', 1: [1, 2, 3, 4]}

# Creating a Dictionary with dict() method
Dict = dict({1: 'Learn', 2: 'with', 3:'phitron'})

# Creating a Dictionary with each item as a Pair
Dict = dict([(1, 'phitron'), (2, 'with')])
#keys must be unique
members = {1001: "John", 1002: "Jane", 1003: "Emily", 1001:
"Max"}print(members)
{1001: 'Max', 1002: 'Jane', 1003: 'Emily'}
```

# Summary



| Data Structure | Ordered | Mutable | Constructor | Example |
|---|---|---|---|---|
| List | Yes | Yes | `[ ]` or `list()` | `[5.7, 4, 'yes', 5.7]` |
| Tuple | Yes | No | `( )` or `tuple()` | `(5.7, 4, 'yes', 5.7)` |
| Set | No | Yes | `{}`* or `set()` | `{5.7, 4, 'yes'}` |
| Dictionary | No | No** | `{ }` or `dict()` | `{'Jun': 75, 'Jul': 89}` |

Thank You