

Data Types



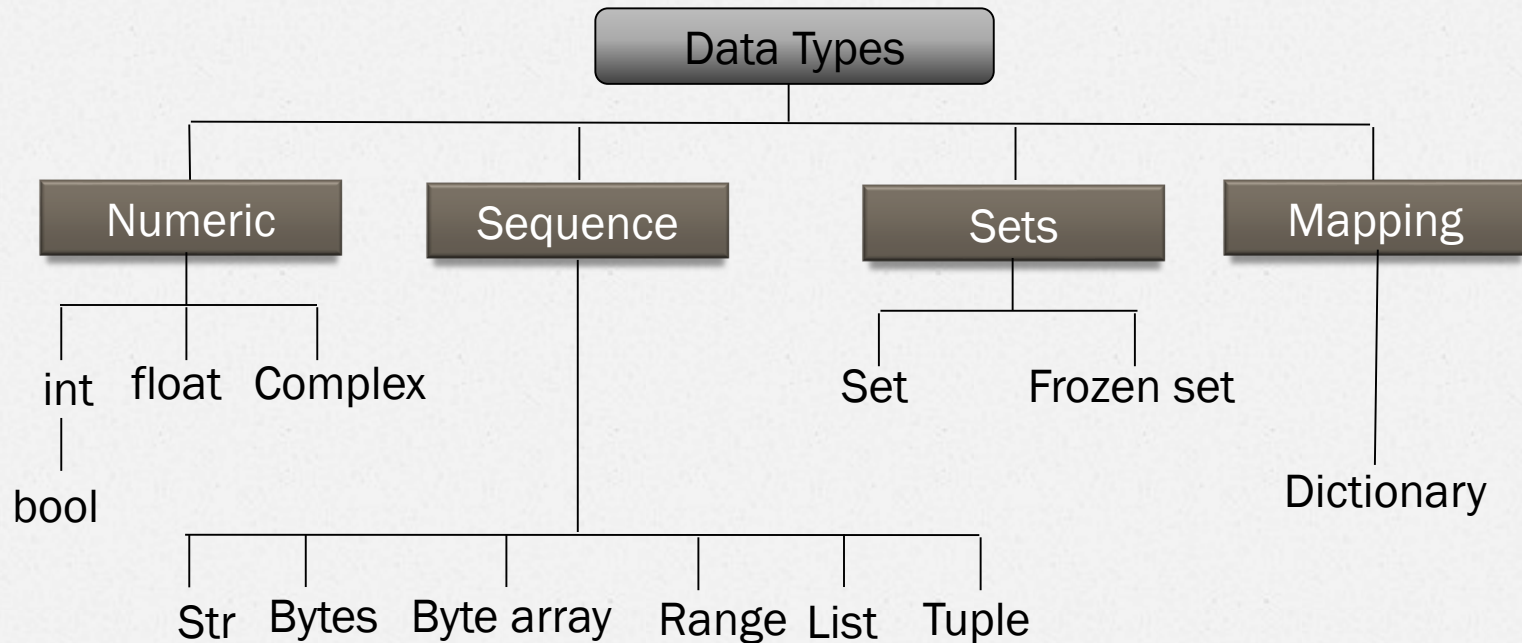
What are data types?

- A data type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data.
- Every value in Python has a data type. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Python allows several data types

- ◌ Int
- ◌ Float
- ◌ Complex
- ◌ bool
- ◌ Strings
- ◌ Bytes
- ◌ Byte array
- ◌ List
- ◌ Tuple
- ◌ range
- ◌ Set
- ◌ Frozen set
- ◌ Dictionary
- ◌ None

Python's data types can be grouped into several classes



Fundamental data Types

- o The first five data types i.e., int, float, complex, bool and str are in-built data types or standard data types.
- o All fundamental data types are immutable.
- o Immutable means can't modify.
- o Mutable means can be modified.



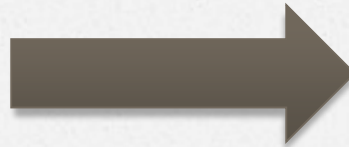
Integer data type

Integer data type

- To hold integral values i.e., whole number e.g., 123, 435.
- 4 ways to represent
 - Decimal form ($x = 10$)
 - Binary form ($x = 0b10$)
 - Octal form ($x = 0o75$)
 - Hexadecimal form ($x = 4F$) — ✕

Example

```
File Edit Forma
p = 10
q = 0b10
r = 0o10
s = 0x10
print(p)
print(q)
print(r)
print(s)
```



```
File Edit She
Python 3.6.
Type "copyr
>>>
= RESTART:
10
2
8
16
>>> |
```

Python provide output in the decimal form.
What if we want output in binary or octal or hex?

Base conversion

- Some built-in functions are used

 bin(x)

Decimal, octal, hex
e.g. bin(15) = 1111

 Oct(x)

Decimal, binary, hex

 Hex(x)

Decimal, binary, octal
e.g. hex(10) = 0xa



Float Data Type

Floating data type

- o E.g., 123.456
- o There is no way to specify float value in binary, octal or hexadecimal.
- o `a = 0x123.45` will through an error

```
>>>  
>>> a = 0b12.4  
SyntaxError: invalid syntax  
>>> a = 0o12.5  
SyntaxError: invalid syntax  
>>> a = 0x3.4  
SyntaxError: invalid syntax
```


Cont....

Exponential form

$f = 1.2e3$

It means $1.2 * 10^3 = 1.2 * 1000 = 1200$

```
>>>  
>>> f = 1.2e3  
>>> print(f)  
1200.0  
>>> |
```



Complex Data Type

Complex data type

- Format is $a + bj$

a is called real part

b is called imaginary part

$$j^2 = -1$$

$$j = \sqrt{-1}$$

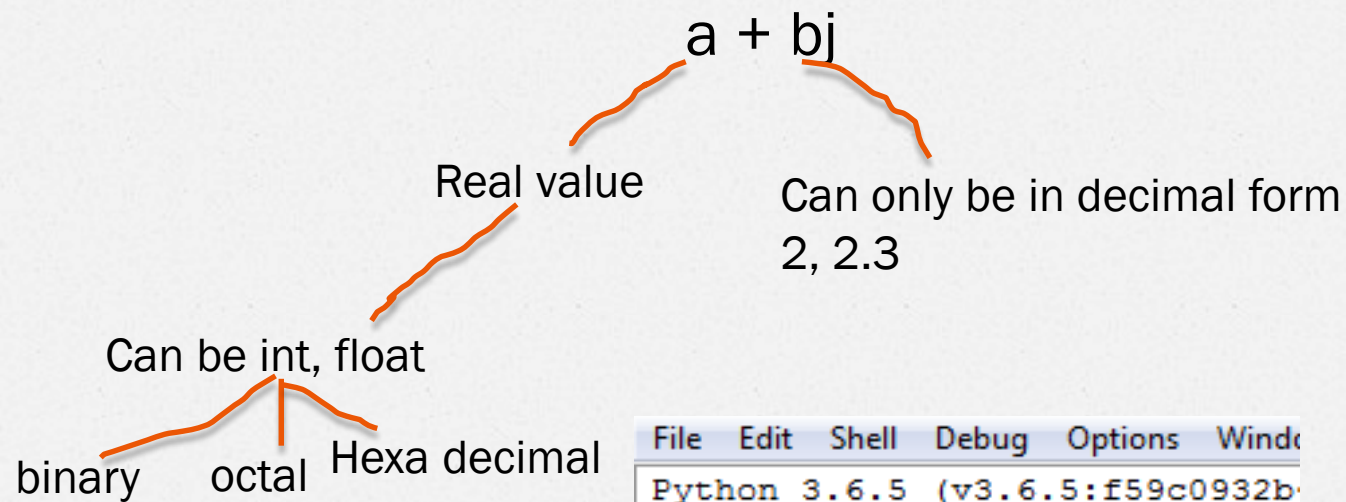
- Use to develop mathematical application or scientific application.

- $10 + 20j$

Only j is valid

$10 + 29i$ will through error

Cont...



```
File Edit Shell Debug Options Window
Python 3.6.5 (v3.6.5:f59c0932b
Type "copyright", "credits" or
>>> a = 0b1111 + 2j
>>> a
(15+2j)
>>> a = 15 + 0b1111j
SyntaxError: invalid syntax
>>> |
```

Operation on complex

```
File Edit Shell Debug Options Win
Python 3.6.5 (v3.6.5:f59c0932
Type "copyright", "credits" c
>>> x = 10 + 20j
>>> x
(10+20j)
>>> type(x)
<class 'complex'>
>>> y = 12.5 + 2.3j
>>> x + y
(22.5+22.3j)
>>> x - y
(-2.5+17.7j)
>>> x.real
10.0
>>> x.imag
20.0
```

Real/imag are not function
they are in built attributes to
get real or imaginary value



Boolean Data Type

Bool data type

- To represent logical values.
- Allowed values of Bool data types

- True

- False

Compulsorily 'T' & 'F' are capital

```
>>> a=True
>>> a
True
>>> a=true
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    a=true
NameError: name 'true' is not defined
>>> |
```

True = 1 & False = 0

True & False are resented as 1 & 0

```
File Edit Shell Debug Options Window
Python 3.6.5 (v3.6.5:f59c0932b4
Type "copyright", "credits" or
>>> True+True
2
>>> True+False
1
>>> False+False
0
>>> a=10
>>> b=20
>>> c=a<b
>>> c
True
>>>
```



Str Data Type

Str data type

- o Any sequence of character known as string.
- o Enclosed in quotes of any type — *single quotation, double quotation and triple quotation (for multiple lines)*.
- o Python strings are immutable(will be discussed).

```
File Edit Format Run O
s = '''john
how are you'''
print(s)
```

```
john
how are you
>>> |
```

File Edit Shell Debug Options Window Help

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46)
Type "copyright", "credits" or "license()" for more info

```
>>> s = john
```

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

```
s = john
```

NameError: name 'john' is not defined

```
>>> s = 'john'
```

```
>>> s
```

```
'john'
```

```
>>> s = "john"
```

```
>>> s
```

```
'john'
```

```
>>> s = "john
```

SyntaxError: EOL while scanning string literal

```
>>> s = '''john
```

```
how are you'''
```

```
>>> s
```

```
'john\nhow are you'
```

```
>>>
```

Should be in quotes

Output in single quote only, even
when input is given in double quote

Triple quote for multiline
In shell it shows \n as line break



Bytes data type

Bytes data type

- Represent a group of byte numbers just like an array.
- Every value should be in the range of 0 to 256
- Bytes data type is immutable.

```
>>> x = [10, 20, 30, 40]
>>> b = bytes(x)
>>> type(b)
<class 'bytes'>
>>>
>>> b[0]          #accessing elements
10
>>> b[1]
20
>>> b[0:3]
b'\n\x14\x1e'
>>>
>>> x = [10, 256, 258, 7]
>>> b = bytes(x)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    b = bytes(x)
ValueError: bytes must be in range(0, 256)
```



Bytearray data type

Bytearray data type

- Same as bytes data type, the only difference is bytearray is mutable

```
>>> x = [10, 20, 30, 40]
>>> b = bytes(x)
>>> b[0] = 50
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    b[0] = 50
TypeError: 'bytes' object does not support item assignment
>>>
>>> x = [10, 20, 30, 40]
>>> b = bytearray(x)
>>> type(b)
<class 'bytearray'>
>>> b[0] = 50
>>> for i in b: print(i)

50
20
30
40
```




List data type

List data type

- Represents a group of comma-separated values of any data type between square brackets.
- Duplication is allowed.
- Lists are mutable i.e., they can be modified.

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07
Type "copyright", "credits" or "license()" for more
>>> a = [10, 10.5, 20, 'john', 20]
>>> print(a)
[10, 10.5, 20, 'john', 20]
>>> a[0] = 99
>>> print(a)
[99, 10.5, 20, 'john', 20]
>>>
```



Tuples data type

Tuples data type

- Tuples are represented as group of comma-separated values of any data type within parenthesis.
- Tuples are same as lists but tuples are immutable.

```
>>> a = (4, 2, 'd', 4.5)
>>> print(a)
(4, 2, 'd', 4.5)
>>> a[0] —————> Accessing element
4
>>> a[0] = 5 —————> Modification is not allowed since tuples are immutable
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    a[0] = 5
TypeError: 'tuple' object does not support item assignment
>>> |
```



Range data type

Range data type

- Represent a sequence of values.
- Immutable
- Can be represented by different forms
 - Form-1 → with one argument
 - E.g.: `range(x)` → represent values from 0 to (x-1)
 - Always Starts from 0
 - Form-2 → with two argument
 - E.g.: `range(x, y)` → represent values from x to (y-1)
 - Starts from x.
 - Form-3 → with three argument
 - E.g.: `range(x, y, z)` → represent values from x to (y-1) with difference of z. Technically word for z is step.

x, y, z that is argument of range always be in integral value

Form-1 [with one argument]

```
>>> r = range(10)
```

```
>>> print(r)
```

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

```
>>>
```

```
>>> r[0]
```

```
0
```

```
>>> r[0:3]
```

```
range(0, 3)
```

```
>>>
```

```
>>> r[0] = 99
```

→ Modification is not allowed (immutable)

```
Traceback (most recent call last):
```

```
File "<pyshell#6>", line 1, in <module>
```

```
    r[0] = 99
```

```
TypeError: 'range' object does not support item assignment
```

```
>>>
```

```
>>> for i in r: print(i)
```

```
0
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

Form-2 [with two argument]

```
>>> r = range(10,30)
>>> r
range(10, 30)
>>> r[5]
15
>>> for i in r: print(i)

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
>>> |
```

Form-3 [with three argument]

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32-bit Intel]
Type "copyright", "credits" or "license()" for more information.
>>> r = range(10,40,5)
>>> for i in r: print(i)          #for i in range(10,40,5): print(i)

10
15
20
25
30
35
>>>
>>> for i in range(2,5,8): print(i)

2
>>>
>>> for i in range(8,1,-3): print(i)

8
5
2
>>>
>>> |
```




Sets data type

Set data type

- o Difference between list and set
 - o In list the order is preserved (important) and duplication is allowed.
 - o In sets, it don't worry about order and don't allow duplicate
- o Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces { }. Items in a set are not ordered.
- o Sets are mutable (i.e., modifiable)
- o Heterogeneous objects are allowed

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.190
Type "copyright", "credits" or "license()" for more information.
>>> s = {10, 20, 30, 10, 20}
>>> s
{10, 20, 30}
>>> s[0]
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    s[0]
TypeError: 'set' object does not support indexing
>>>
>>> s.add('john')
>>> s
{'john', 10, 20, 30}
>>>
>>> s.remove(20)
>>> s
{'john', 10, 30}
>>>
```

- Indexing or slice operator is not allowed since there is no guarantee in sets that which element is at first position or which at second bcuz order is not preserved



Frozenset data type

frozenset data type

- Exactly same as set data type except it is immutable

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900
Type "copyright", "credits" or "license()" for more information.
>>> s = {10,20,30,40}
>>> fs = frozenset(s)
>>> type(fs)
<class 'frozenset'>
>>>
>>> fs
frozenset({40, 10, 20, 30})
>>>
>>> fs.add(50) #through error bcuz it is immutable (Unchangeable)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    fs.add(50) #through error bcuz it is immutable (Unchangeable)
AttributeError: 'frozenset' object has no attribute 'add'
>>> |
```

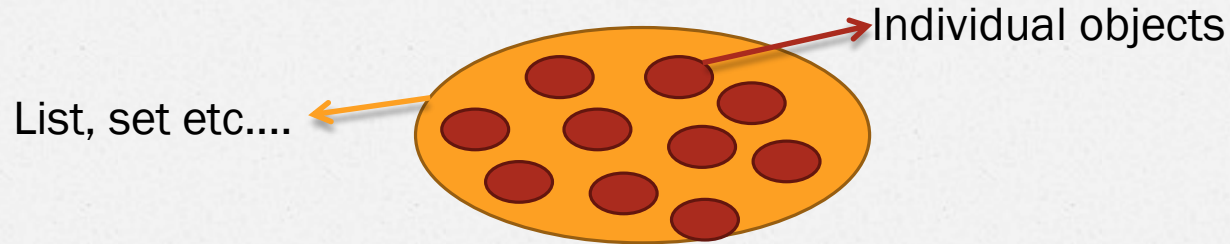


Dictionary data type

dict data type

o Till yet we have studied about
bytes, bytearray, list, tuple, set, frozenset, range

A group of individual elements/objects, there is
no relation between them.

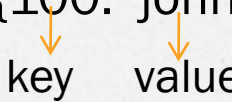


What if I want some relation between objects like
Roll no → name (relation between roll no. and name of stu.)
Word → meaning (relation between word and its meaning)

Cont...

- Dictionary is collection of entries in which each entry contains a key and a corresponding value.

Key	Value
100	John
200	smith

- Duplication is keys are not allowed but values can be duplicate
- Represented as `d = {100: 'john', 200: 'smith'}`

key value
- Key and values can be heterogeneous
- `d = { }` #an empty dictionary then what about empty set?
- Empty set can be represented as: `s = set()`

Cont...



File Edit Shell Debug Options Window Help

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32-bit Intel]

Type "copyright", "credits" or "license()" for more information.

```
>>> d = {100: 'john', 200: 'smith'}
```

```
>>> type(d)
```

```
<class 'dict'>
```

```
>>>
```

```
>>> d
```

```
{100: 'john', 200: 'smith'}
```

```
>>>
```

```
>>> s = {}          #empty set or dictionary?
```

```
>>> type(s)
```

```
<class 'dict'>
```

```
>>>
```

```
>>> s = set()
```

```
>>> type(s)
```

```
<class 'set'>
```

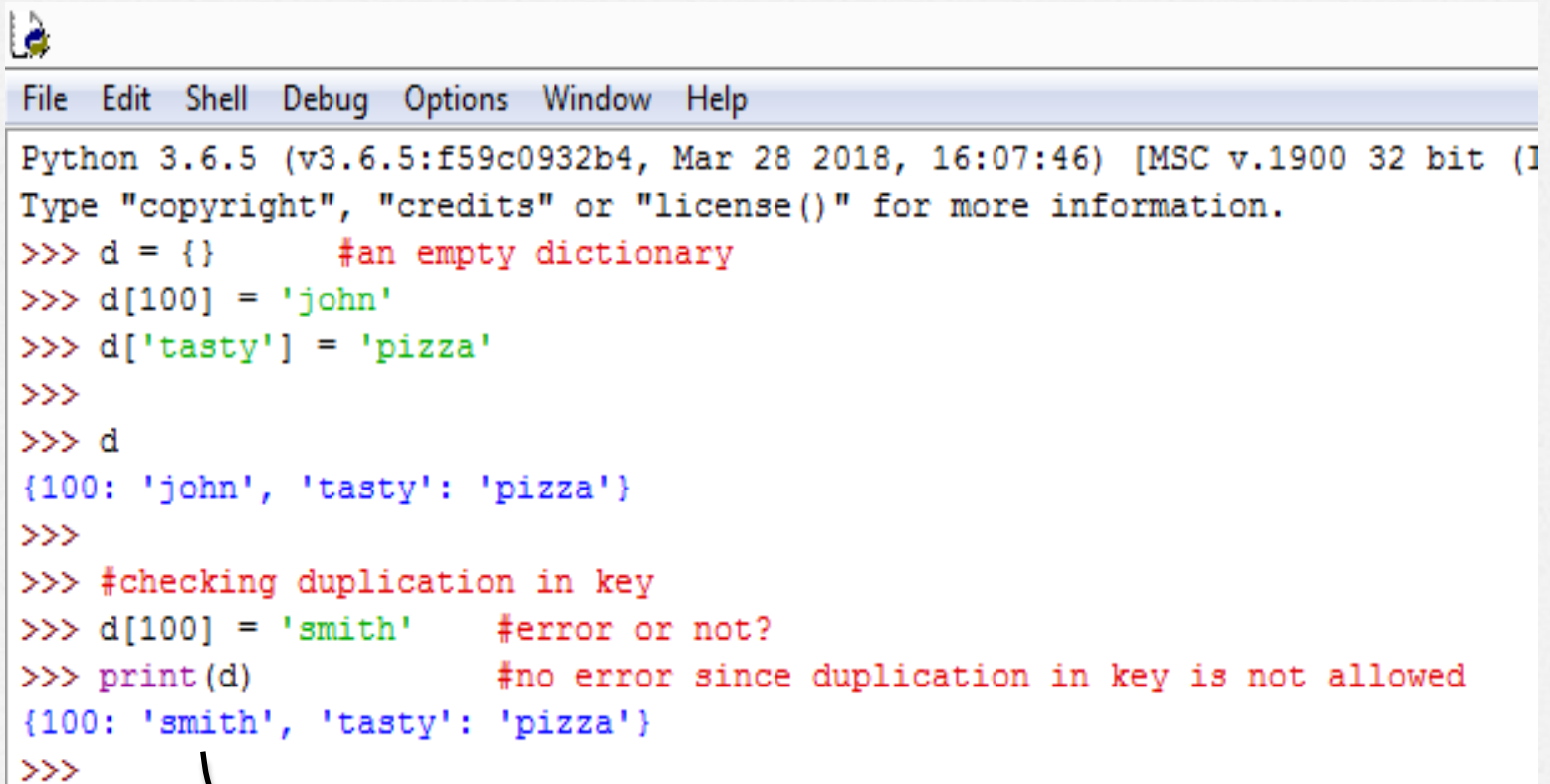
```
>>> s
```

```
set()
```

```
>>>
```


Adding elements to dictionary

- Dictionary is mutable so we can allowed to add or remove objects.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (I
Type "copyright", "credits" or "license()" for more information.
>>> d = {}          #an empty dictionary
>>> d[100] = 'john'
>>> d['tasty'] = 'pizza'
>>>
>>> d
{100: 'john', 'tasty': 'pizza'}
>>>
>>> #checking duplication in key
>>> d[100] = 'smith'    #error or not?
>>> print(d)           #no error since duplication in key is not allowed
{100: 'smith', 'tasty': 'pizza'}
>>>
```

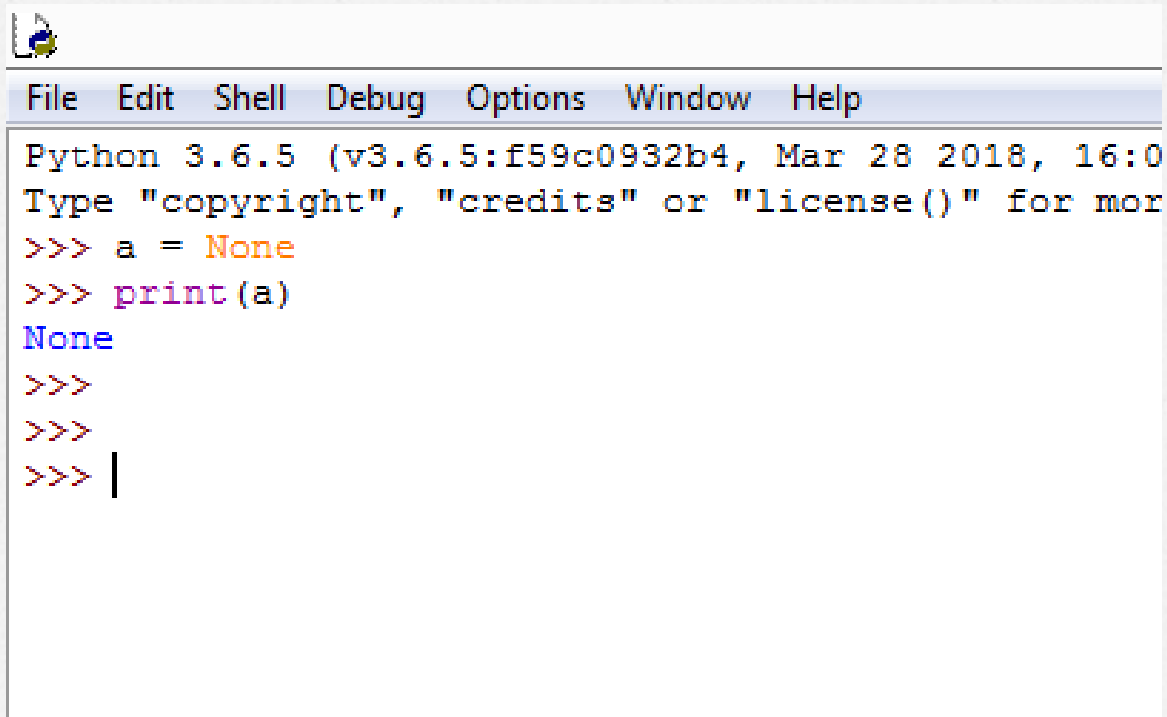
Old value(john) replaced by new value



None data type

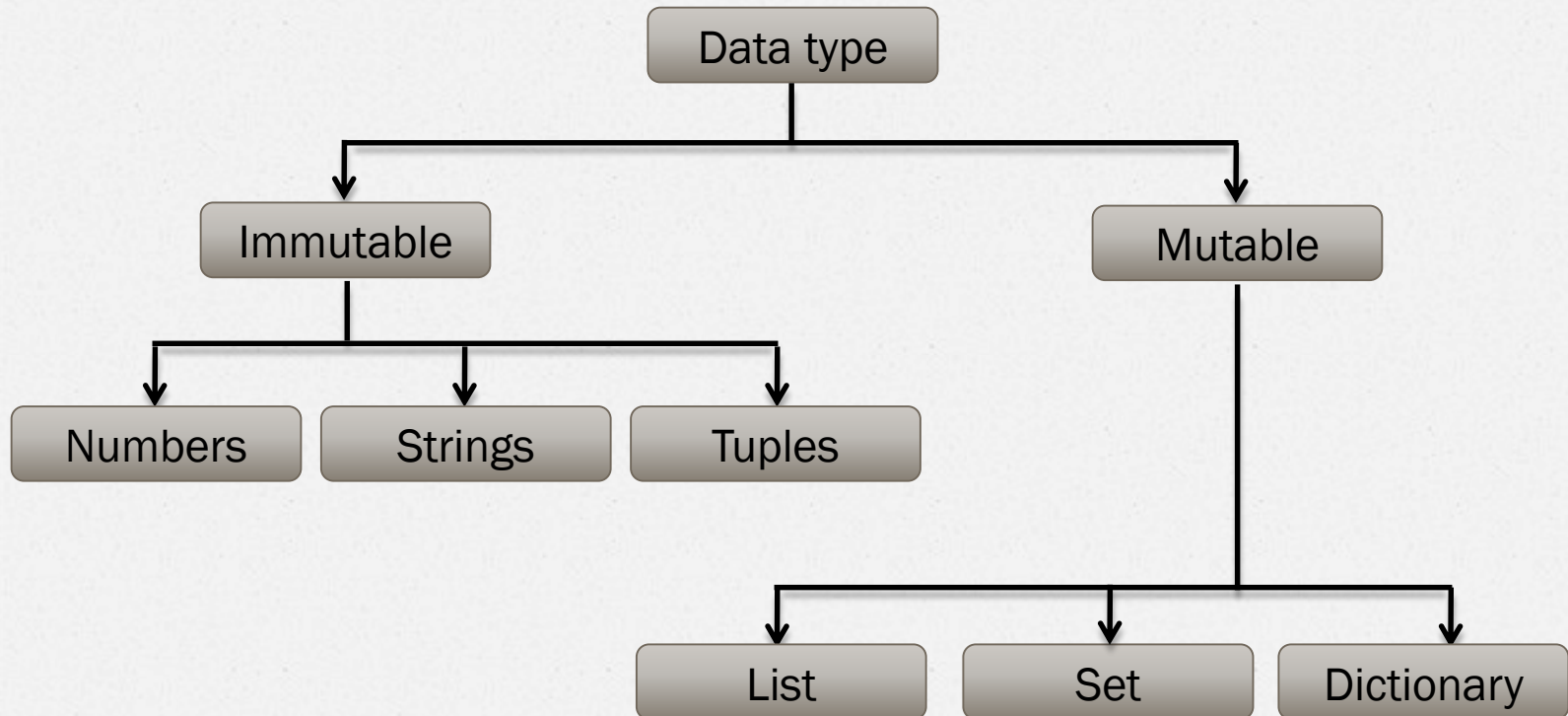
None data type

- None means nothing or no value associated.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:0
Type "copyright", "credits" or "license()" for mor
>>> a = None
>>> print(a)
None
>>>
>>>
>>> |
```


Python data type classified into:



Class	Immutable?	Mutable?
Int	✓	
Float	✓	
Complex	✓	
Bool	✓	
Str	✓	
Bytes	✓	
Bytearray		✓
Range	✓	
List		✓
Tuple	✓	
Set		✓
Frozenset	✓	
Dictionary		✓