# **Python Data Types**

(source: https://www.programiz.com/python-programming)

### **[Python Numbers](https://www.programiz.com/python-programming/variables-datatypes#number)**

### **[Python List](https://www.programiz.com/python-programming/variables-datatypes#list)**

### **[Python Tuple](https://www.programiz.com/python-programming/variables-datatypes#tuple)**

### **[Python Strings](https://www.programiz.com/python-programming/variables-datatypes#string)**

### **[Python Set](https://www.programiz.com/python-programming/variables-datatypes#set)**

### **[Python Dictionary](https://www.programiz.com/python-programming/variables-datatypes#dic)**

### **[Conversion between data types](https://www.programiz.com/python-programming/variables-datatypes#con)**

## **Data types in Python**

### Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

### There are various data types in Python. Some of the important types are listed below.

### **Python Numbers**

Integers, floating point numbers and complex numbers falls under [Python numbers](https://www.programiz.com/python-programming/numbers) category. They are defined as int, float and complex class in Python.

We can use the type() function to know which class a variable or a value belongs to and the isinstance() function to check if an object belongs to a particular class.

a = 5

print(a, "is of type", type(a))

a = 2.0

print(a, "is of type", type(a))

a = 1+2j

print(a, "is complex number?", isinstance(1+2j,complex))

*Integers can be of any length, it is only limited by the memory available.*

A floating point number is accurate up to 15 decimal places. Integer and floating points are separated by decimal points. 1 is integer, 1.0 is floating point number.

Complex numbers are written in the form, x + yj, where x is the real part and y is the imaginary part. Here are some examples.

>>> a = 1234567890123456789  
>>> a  
1234567890123456789  
>>> b = 0.1234567890123456789  
>>> b  
0.12345678901234568  
>>> c = 1+2j  
>>> c  
(1+2j)

Notice that the float variable b got truncated.

### **Python List**

[List](https://www.programiz.com/python-programming/list) is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type.

Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [ ].

>>> a = [1, 2.2, 'python']

We can use the slicing operator [ ] to extract an item or a range of items from a list. Index starts from 0 in Python.

a = [5,10,15,20,25,30,35,40]

# a[2] = 15

print("a[2] = ", a[2])

# a[0:3] = [5, 10, 15]

print("a[0:3] = ", a[0:3])

# a[5:] = [30, 35, 40]

print("a[5:] = ", a[5:])

Lists are mutable, meaning, value of elements of a list can be altered.

>>> a = [1,2,3]  
>>> a[2]=4  
>>> a  
[1, 2, 4]

### **Python Tuple**

[Tuple](https://www.programiz.com/python-programming/tuple) is an ordered sequence of items same as list.The only difference is that tuples are immutable. Tuples once created cannot be modified.

Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.

It is defined within parentheses () where items are separated by commas.

>>> t = (5,'program', 1+3j)

We can use the slicing operator [] to extract items but we cannot change its value.

t = (5,'program', 1+3j)

# t[1] = 'program'

print("t[1] = ", t[1])

# t[0:3] = (5, 'program', (1+3j))

print("t[0:3] = ", t[0:3])

# Generates error

# Tuples are immutable

t[0] = 10

### **Python Strings**

[String](https://www.programiz.com/python-programming/string) is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, ''' or """.

>>> s = "This is a string"  
>>> s = '''a multiline

Like list and tuple, slicing operator [ ] can be used with string. Strings are immutable.

s = 'Hello world!'

# s[4] = 'o'

print("s[4] = ", s[4])

# s[6:11] = 'world'

print("s[6:11] = ", s[6:11])

# Generates error

# Strings are immutable in Python

s[5] ='d'

### **Python Set**

[Set](https://www.programiz.com/python-programming/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.

a = {5,2,3,1,4}

# printing set variable

print("a = ", a)

# data type of variable a

print(type(a))

We can perform set operations like union, intersection on two sets. Set have unique values. They eliminate duplicates.

>>> a = {1,2,2,3,3,3}  
>>> a  
{1, 2, 3}

Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [] does not work.

>>> a = {1,2,3}  
>>> a[1]  
Traceback (most recent call last):  
 File "<string>", line 301, in runcode  
 File "<interactive input>", line 1, in <module>  
TypeError: 'set' object does not support indexing

### **Python Dictionary**

[Dictionary](https://www.programiz.com/python-programming/dictionary) is an unordered collection of key-value pairs.

It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.

In Python, dictionaries are defined within braces {} with each item being a pair in the form key:value. Key and value can be of any type.

>>> d = {1:'value','key':2}  
>>> type(d)  
<class 'dict'>

We use key to retrieve the respective value. But not the other way around.

d = {1:'value','key':2}

print(type(d))

print("d[1] = ", d[1]);

print("d['key'] = ", d['key']);

# Generates error

print("d[2] = ", d[2]);

### **Conversion between data types**

We can convert between different data types by using different type conversion functions like int(), float(), str() etc.

>>> float(5)  
 5.0

Conversion from float to int will truncate the value (make it closer to zero).

>>> int(10.6)  
10  
>>> int(-10.6)  
-10

Conversion to and from string must contain compatible values.

>>> float('2.5')  
2.5  
>>> str(25)  
'25'  
>>> int('1p')

Traceback (most recent call last):  
 File "<string>", line 301, in runcode  
 File "<interactive input>", line 1, in <module>  
 ValueError: invalid literal for int() with base 10: '1p'

We can even convert one sequence to another.

>>> set([1,2,3])  
{1, 2, 3}  
>>> tuple({5,6,7})  
(5, 6, 7)  
>>> list('hello')  
['h', 'e', 'l', 'l', 'o']

To convert to dictionary, each element must be a pair

>>> dict([[1,2],[3,4]])  
{1: 2, 3: 4}  
>>> dict([(3,26),(4,44)])  
{3: 26, 4: 44}

Extras,,,,

**Swapping**

>>> a=10

>>> b=20

>>> a

10

>>> b

20

**>>> a,b=b,a**

>>> a

20

>>> b

10