Bytewise Fellowship Program

# DATA SCIENCE

Task #5

BWT- Data Science (Group1)

Submitted to: Mahrukh Khan

Submitted by: Usama Malik

# Task: Data Structures and Sequences

In Python, frequently used sequences include tuples, lists, and dictionaries.

## Tuple

A tuple has a fixed length, meaning that once an object is assigned to a tuple, it cannot be changed. Tuples are declared using parentheses and separated by commas.

**Example:**

```
In [2]: tup = (4, 5, 6)

In [3]: tup

Out[3]: (4, 5, 6)
```

We can convert a sequence to a tuple by invoking the `tuple` function:

```
In [6]: tuple([4, 0, 2])

Out[6]: (4, 0, 2)

In [7]: tup = tuple('string')

In [8]: tup

Out[8]: ('s', 't', 'r', 'i', 'n', 'g')
```

## Unpacking

If you try to assign a tuple to multiple variables, Python will attempt to unpack the values on the right-hand side of the equals sign:

**Example:**

```
In [20]: tup = (4, 5, 6)

In [21]: a, b, c = tup

In [22]: b

Out[22]: 5
```

## List

The values of a list can be changed, it has a variable length, and it can be modified.

**Example:**

```
In [48]: gen = range(10)

In [49]: gen

Out[49]: range(0, 10)

In [50]: list(gen)

Out[50]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

## Concatenating and Combining Lists

Similar to tuples, adding two lists together with + concatenates them:

**Example:**

```
In [63]: [4, None, "foo"] + [7, 8, (2, 3)]

Out[63]: [4, None, 'foo', 7, 8, (2, 3)]
```

## Sorting

You can sort a list in place (without creating a new object) by calling its sort function:

Example:

```
In [67]: a = [7, 2, 5, 1, 3]

In [68]: a.sort()

In [69]: a

Out[69]: [1, 2, 3, 5, 7]
```

## Slicing

You can select sections of most sequence types by using slice notation, which in its basic form consists of start:stop passed to the indexing operator []:

**Example:**

```
In [73]: seq = [7, 2, 3, 7, 5, 6, 0, 1]
In [74]: seq[1:5]

Out[74]: [2, 3, 7, 5]
```

## Dictionary

A dictionary is a collection of key-value pairs where each key is associated with a value. We can retrieve, modify, insert, and delete these values. Dictionaries are created using curly braces {} and key-value pairs are separated by commas.

**Example:**

```
In [83]: empty_dict = {}

In [84]: d1 = {"a": "some value", "b": [1, 2, 3, 4]}

In [85]: d1

Out[85]: {'a': 'some value', 'b': [1, 2, 3, 4]}
```

## Set

A set is an unordered collection of unique elements. There are two ways to create a set: using the set function or with curly braces.

**Example:**

```
In [124]: set([2, 2, 2, 1, 3, 3])

Out[124]: {1, 2, 3}

In [125]: {2, 2, 2, 1, 3}
```

## Function

As a rule of thumb, if you anticipate needing to repeat the same or very similar code more than once, it may be worth writing a reusable function.

**Example:**

```
In [174]: def my_function(x, y):

   ...:     return x + y

   ...:

In [175]: result = my_function(1, 2)

In [176]: result

Out[176]: 3
```

## Errors and Exception Handling

For example, Python’s float function is capable of casting a string to a floating-point number, but it fails with a ValueError on improper inputs:

**Example**

```
In [224]: float("1.2345")

Out[224]: 1.2345

In [225]: float("something")

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

ValueError: could not convert string to float: 'something':
```

## Files and the Operating System

It is relatively straightforward to handle files in Python, which is one reason Python is popular for text and file processing. To open a file for reading or writing, use the built-in open function with either a relative or absolute file path or an optional file encoding.