How to Make the Best Use of Live Sessions

- Please **log in 10 mins before** the class starts and check your internet connection to avoid any network issues during the LIVE session
- All participants will be on mute, by default, to avoid any background noise. However, you will be unmuted by instructor if required. Please use the "**Questions**" tab on your webinar tool to interact with the instructor at any point during the class
- Feel free to ask and answer questions to make your learning interactive. Instructor will address your queries at the end of on-going topic
- Raise a ticket through your LMS in case of any queries. Our dedicated support team is available 24 x 7 for your assistance
- **Your feedback is highly appreciated. Please share your valuable feedback after each class to help us enhance your learning experience**

edureka!

edureka!

Python Programming Certification Course

COURSE OUTLINE
MODULE 02

edureka!

Sequences and File Operations

Topics

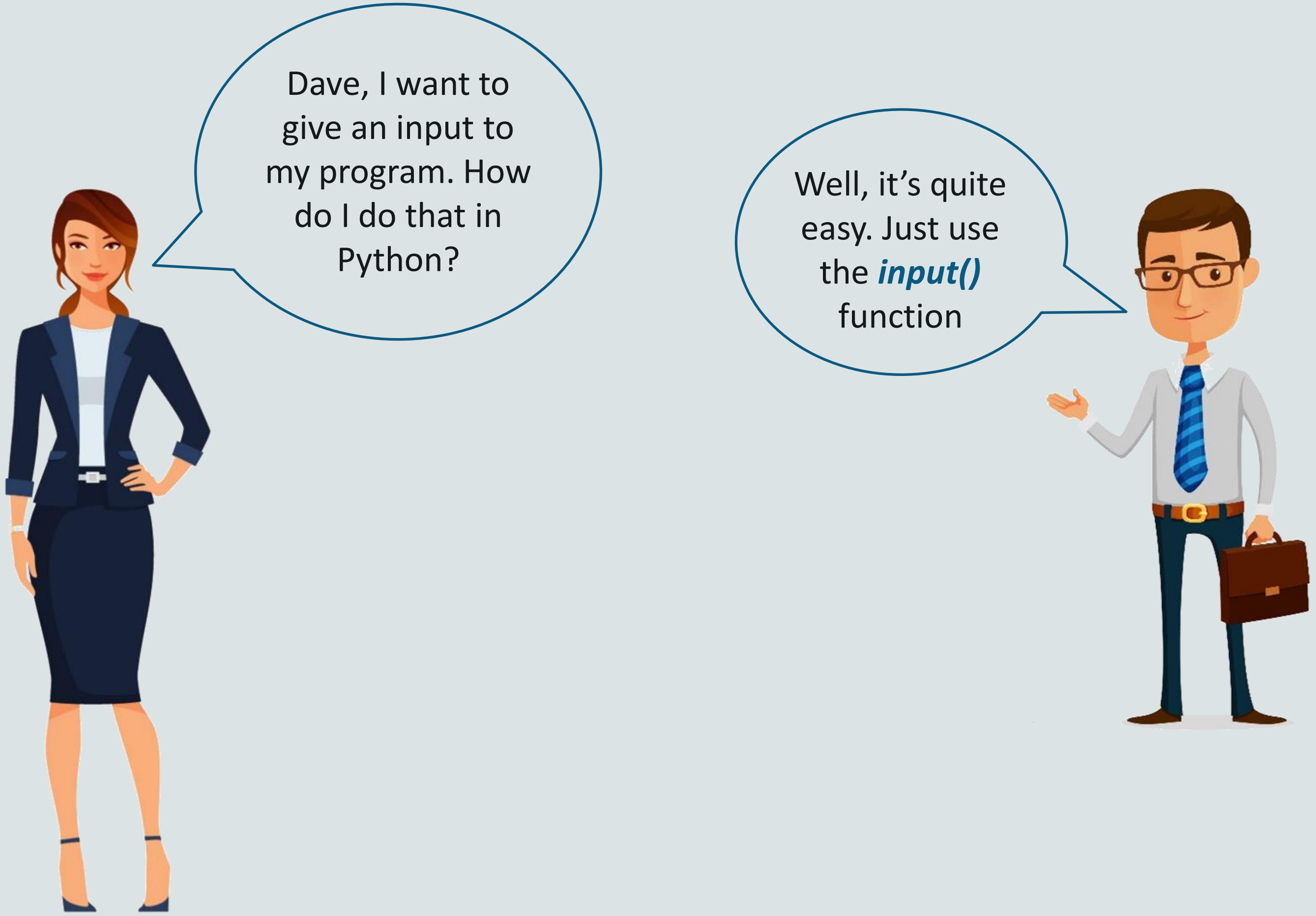
Following topics are covered in this module:

- Reading keyboard input in Python
- File input/output operations in Python
- File objects in Python
- Types of sequences and its operations in Python:
    - Lists
    - Tuples
    - Strings
    - Sets
    - Dictionaries

Objectives

After completing this module, you should be able to:

- Understand operations performed on files
- Learn what sequences are
- Execute sequence operations
- Understand types of sequences in Python


Dave, I want to give an input to my program. How do I do that in Python?
Well, it's quite easy. Just use the ***input()*** function

Reading Keyboard Input

Reading Keyboard Input

Python provides a built-in function ***input*** () to read a line of text from the standard function

```
user_input=input('Enter Your value')
print('The value entered by user:',user_input)
print('The datatype of the value entered by the user:',type(user_input))
```

Output

```
Enter Your value10
The value entered by user: 10
The datatype of the value entered by the user: <class 'str'>
```

Although the user entered an integer, the data type shown is string. How is it possible? Is the interpreter working right?

edureka!

Reading Keyboard Input – *eval()* Function

Reading Keyboard Input

By default, all the inputs entered by users are considered as **string**. Python provides a built-in function ***eval()*** to retain the original data type of the entered value

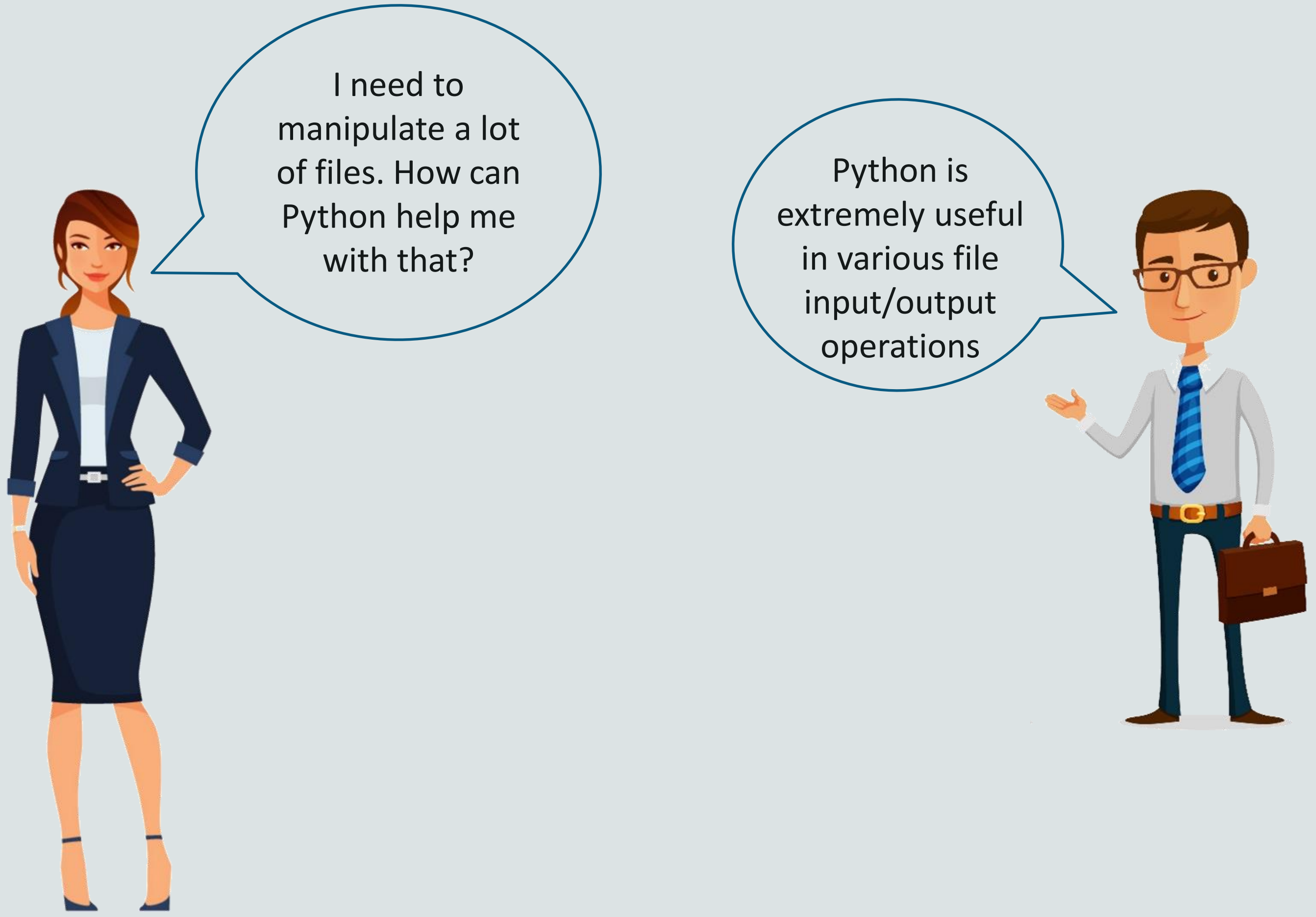
```
user_input=input('Enter Your value')
print('The value entered by user:',user_input)
print('The datatype of the value entered by the user:',type(eval(user_input)))
```

Output

```
Enter Your value10
The value entered by user: 10
The datatype of the value entered by the user: <class 'int'>
```

Do we have any other method to get back the original data type?

```
user_input=int(input('Enter Your value'))
print('The value entered by user:',user_input)
print('The datatype of the value entered by the user:',type(user_input))
```


I need to manipulate a lot of files. How can Python help me with that?
Python is extremely useful in various file input/output operations

edureka!

Python Files Input/output Operations

Opening and closing files

Writing and reading files

Renaming files

Opening and Closing Files

Before reading and writing any data into a file, it is important to learn how to open and close a file

Opening files

Unless you open a file, you can not write anything in a file or read anything from it

Closing files

Once you are done with reading or writing, close the file

open() Function

- You can open Files using Python's built-in *open()* function

```
file_Object=open(file_name,[access_mode])
```

- Here are the parameter details:

  file_name: The file_name argument is a **string** value that contains the name of the file that you want to access

  access_mode: The access_mode determines the mode in which the file has to be opened, i.e., read, write, append etc.

edureka!

open() Function – Access Modes

Modes	Description
r	This is the default mode and is used for opening a file in read only mode
rb	opens a file to read only in binary form
r+	opens a file for both reading and writing
rb+	opens a file to read and write in binary format
w	opens a file in write only mode. If the file exists, it overwrites the same or else creates a new one.
wb	opens a file for writing only in binary format. If the file exists, it overwrites the same or else creates a new one.

edureka!

open() Function – Access modes (Cont.)

Modes	Description
a	opens a file to append
ab	opens a file to append in binary format
a+	opens a file to append and read
ab+	opens a file to append and read in binary format
w+	opens a file to read and write
wb+	opens a file to read and write in binary format

edureka!

Writing Files

fileObject.write(string)

The ***write()*** method does not add a newline character **'\n'** to the end of the string

The ***write()*** method writes content in an open file.

Note:- Python strings can have binary data and not just text

edureka!

Reading Files

fileObject.read([count])

The ***read()*** method reads a string from an open file

Note :- It is important to note that Python strings can have binary data apart from text data

Renaming Files

```
os.rename(current_file_name, new_file_name)
```

The ***rename()*** method takes two arguments, the **current filename** and the **new filename**

rename() is the method from ***os*** module. We are going to learn ***os*** module in detail in **Module 4**

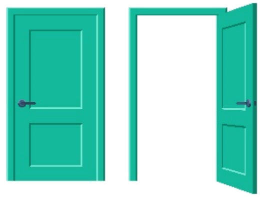
Deleting Files

```
os.remove(file_name)
```

You can use the ***remove()*** method to delete files by supplying the name of the file to be deleted as an argument

remove() is the method from ***os*** module

Closing Files



file.close()

The ***close()*** method closes the opened file

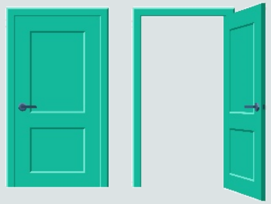
Note :- A closed file cannot be read or written any more

Note :- Python automatically closes a file when the reference object of a file is reassigned to another file

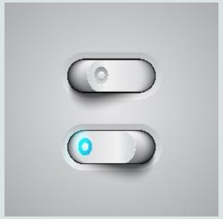
edureka!

File Object Attributes

- After opening a file, various information related to that file can be obtained using the file object
- Here is a list of all attributes related to file object:



file.closed



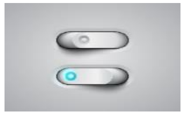
file.mode

file.name

File Object Attributes (Cont.)

file.name

The attribute ***name*** returns the name of File



file.mode

The attribute ***mode*** returns the mode in which the file is opened

file.closed

The attribute ***closed*** returns True if the file is closed.

File Object Methods

file.seek() → The ***seek(offset[, from])*** method changes the current file position

file.tell() → The ***tell()*** method can be used to find the current position within the file and the next read or write will occur at that many bytes from the beginning of the file

Demo 1: User Input and File Handling

Note: Refer to Module-2 Demo1 File Handling file on LMS for all the steps in detail

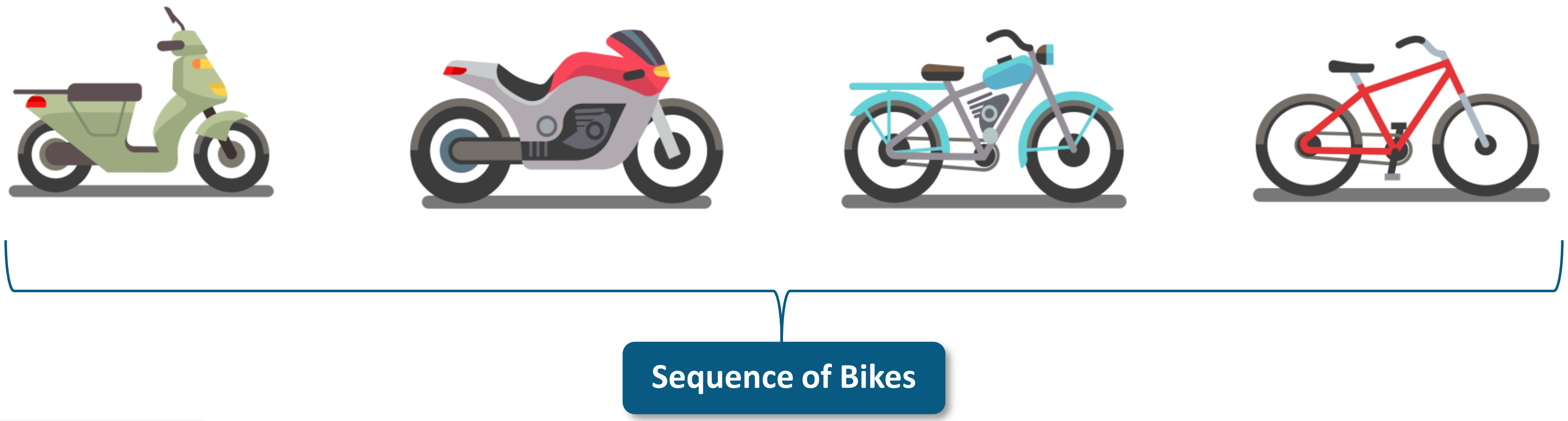
Dave, I have different values of different data types. How do I deal with them?

Python provides the best solution for what you are looking for

Sequences and Its Various Operations

What are Sequences?

Sequences are containers with items that are accessible by indexing or slicing. The built-in ***len()*** function is used to find the number of items in a container



Sequence Operations

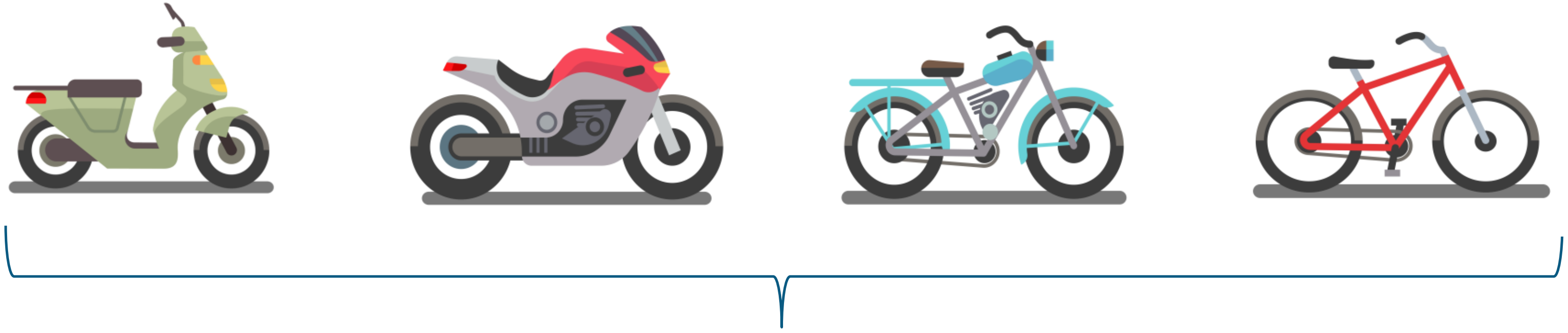
Concatenation

Repetition

Membership Testing

Slicing

Indexing

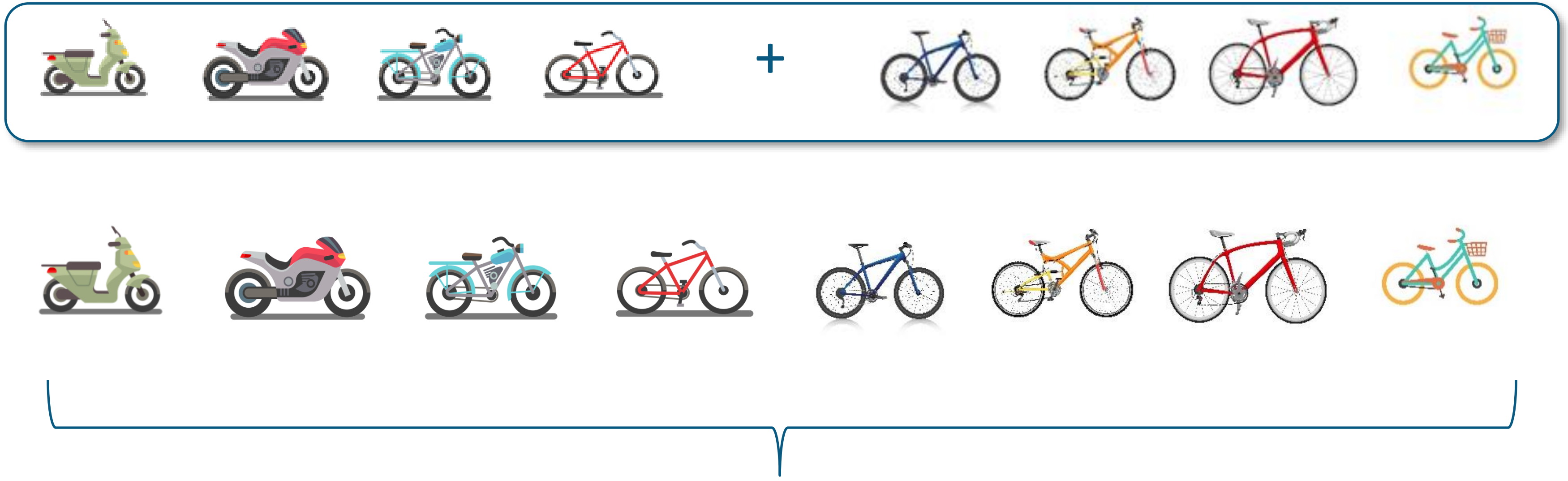


Sequence of Bikes

edureka!

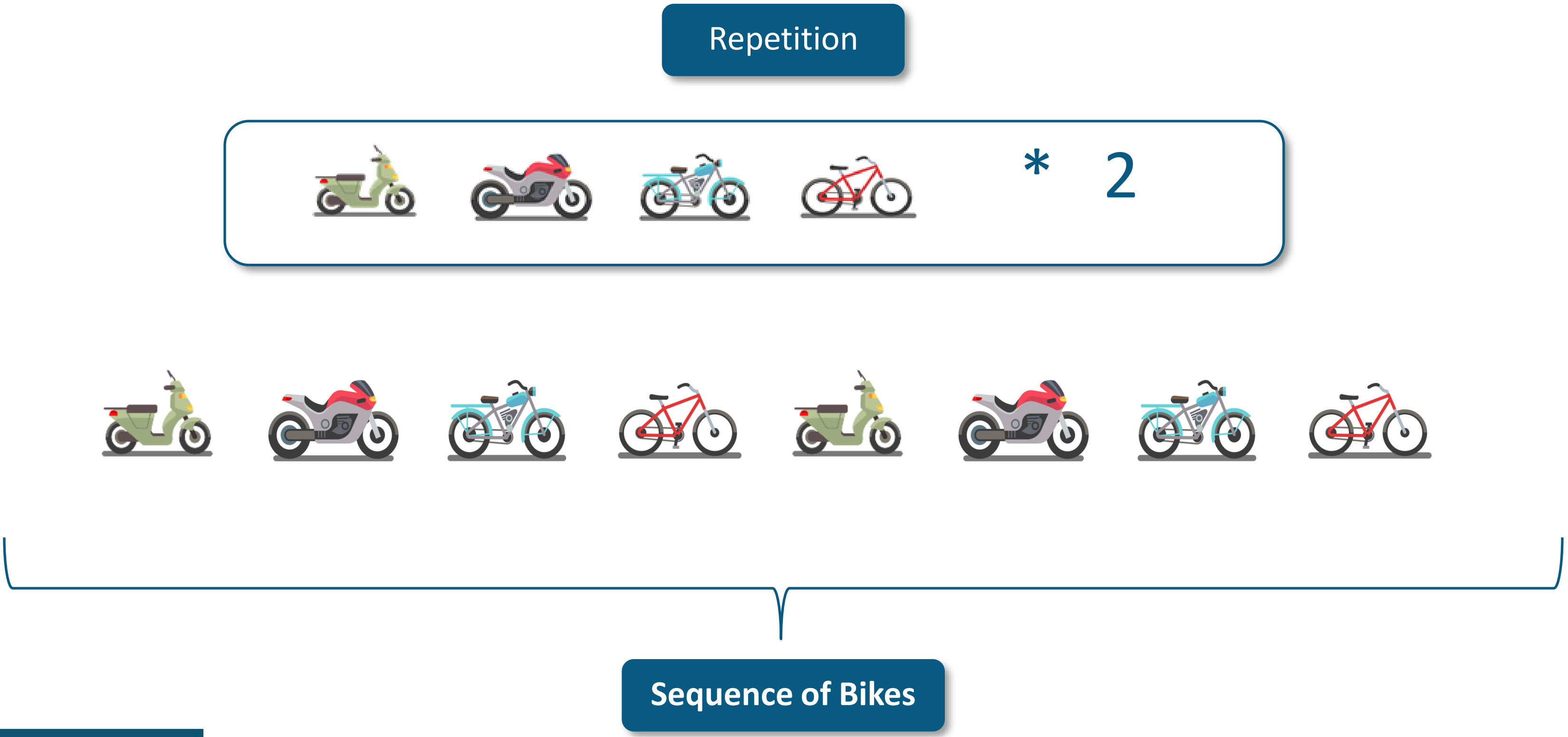
Sequence Concatenation

Concatenation



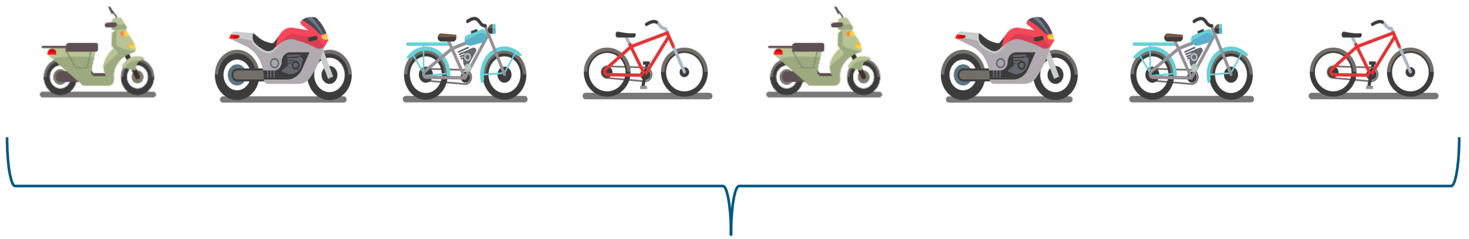
Sequence of Bikes

Sequence Repetition



Sequence Membership Testing

Membership Testing

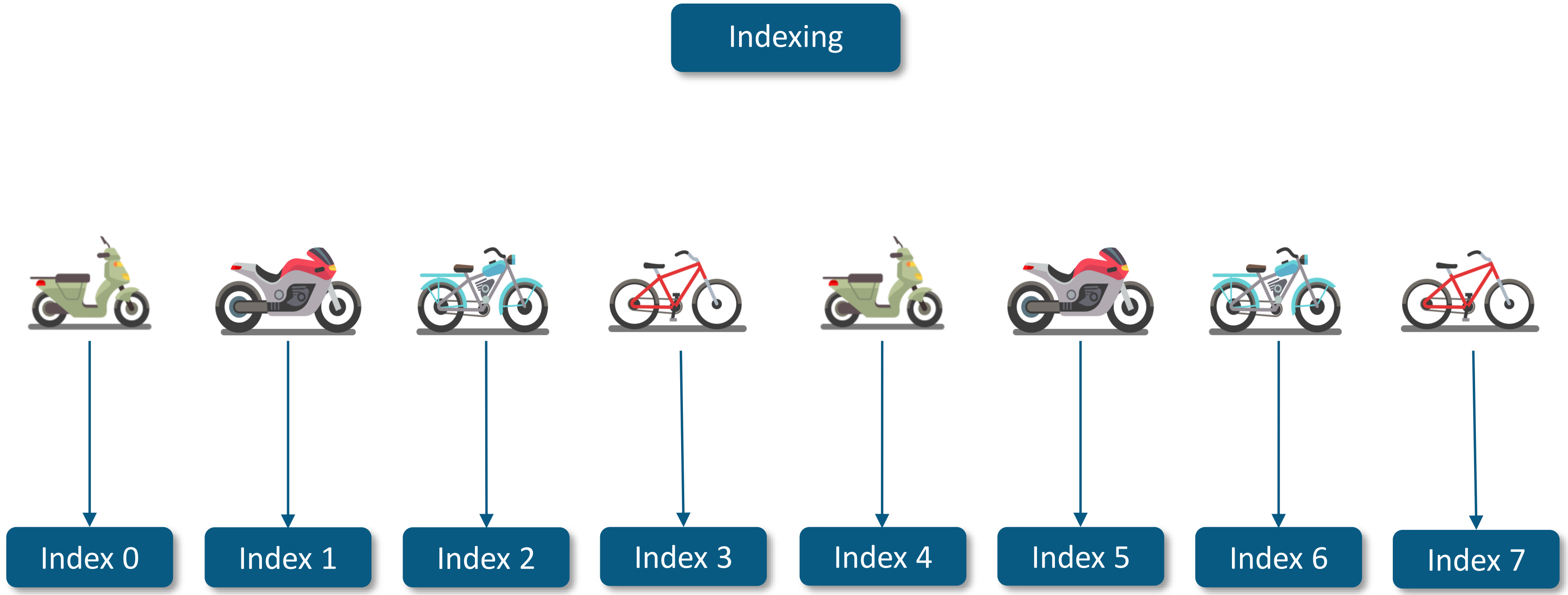


Sequence of Bikes

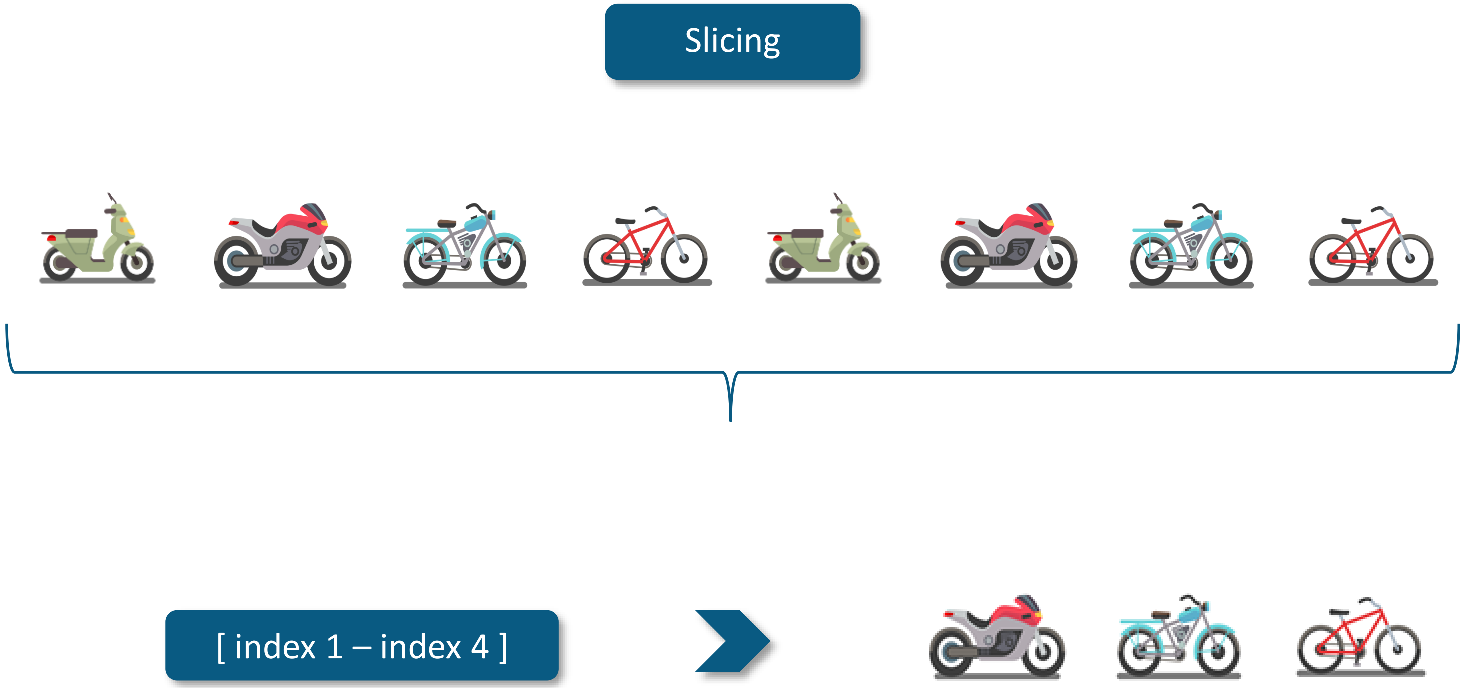
Not a member

Is a member

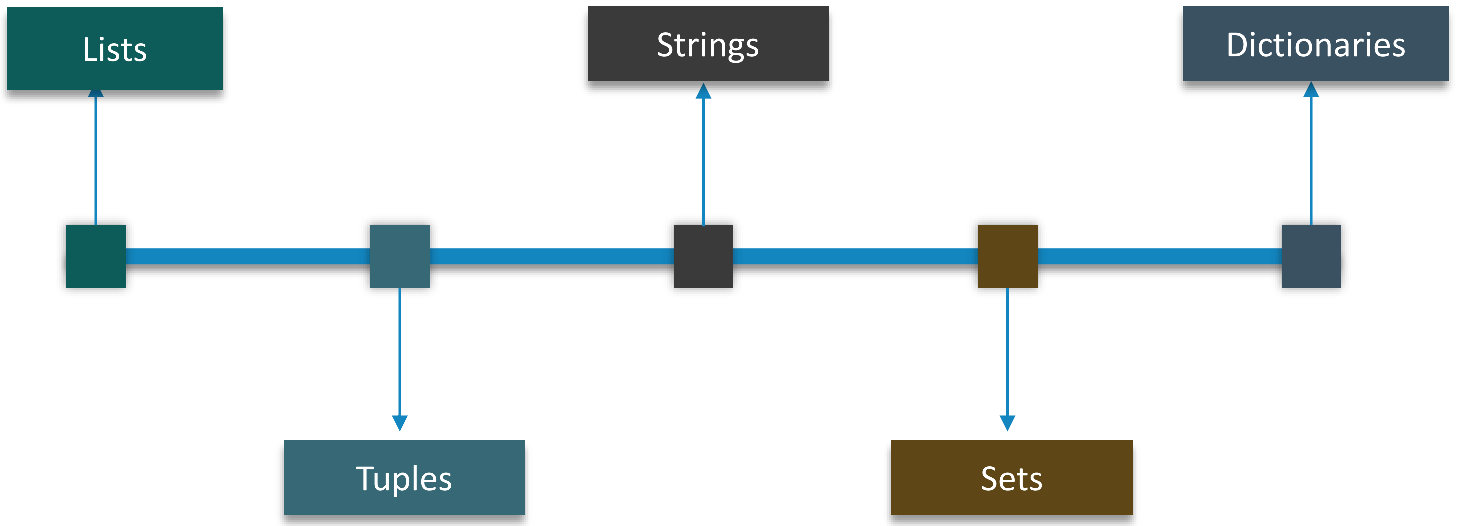
Sequence Indexing



Sequence Slicing



Types of Sequences In Python



edureka!

Lists

Lists

Tuples

Strings

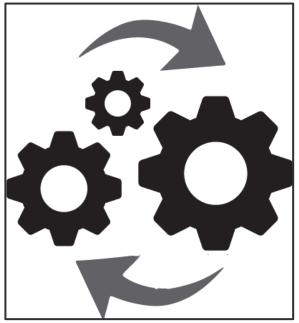
Sets

Dictionaries

List is the most versatile datatype available in Python, which can be written as a list of comma-separated values (items) between square brackets

Example:
list=[“Marketing”,”Sales”,8,11]

Operations



- Slicing
- Updating List
- Deleting List Elements
- List Length
- Concatenation
- Repetition

edureka!

When to Use Lists?

If you have a collection of data that needs both sequential and random access

When you have to deal with values which can be changed

Demo 2: List Operations

Note: Refer to Module-2 Demo2 file (Sequences-Lists) on LMS for all the steps in detail

edureka!

Tuples

Lists

Tuples

Strings

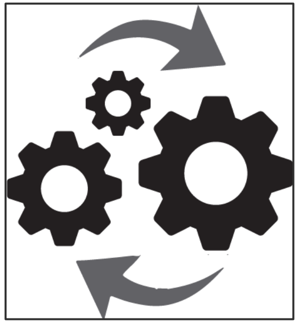
Sets

Dictionaries

A Tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists

Example: tuple=("Marketing","Sales")

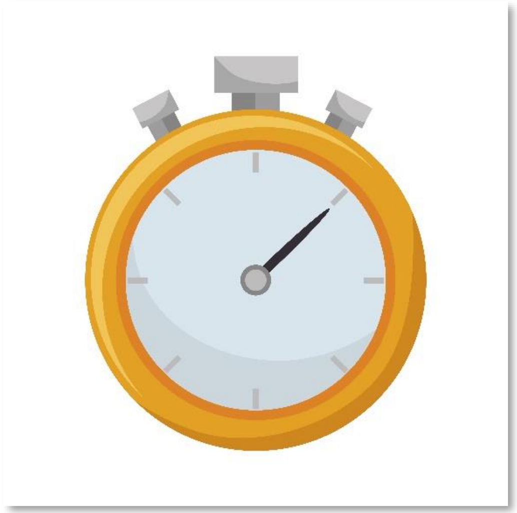
Operations



- Slicing
- Updating Tuple
- Deleting Tuple
- Tuple Length
- Concatenation
- Repetition

When to Use Tuples?

When you need to complete the task in a short time (Tuple has less execution time)



Where you have to deal with values which can not be changed

When to Use Tuples? (Cont.)

I have a constant set of values and I have to iterate through them, what to do?

John

A Tuple is a collection of constant values, and the speed of execution of tuple is faster than other sequences. So for John, tuple will be the best choice

Demo 3: Tuple Operations

Note: Refer to Module-2 Demo3 file (Sequences-Tuples) on LMS for all the steps in detail

edureka!

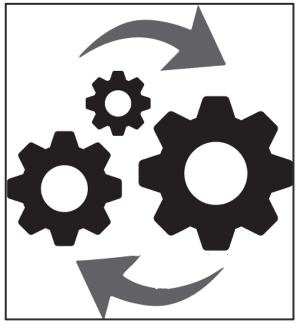
Strings

- Lists
- Tuples
- **Strings**
- Sets
- Dictionaries

We can create them simply by enclosing characters in quotes

Example: string="Python"

Operations



Slicing	String[range]
Updating	String[range] + 'x'
Concatenation	String 1 + String 2
Repetition	String 1 * x
Membership	In, not in
Reverse	String [:-1]

edureka!

String Formatting Operators

Operators	Conversion
%c	character
%i	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer lower case letters
%e	exponential notation with lower case 'e'
%f	floating point real number
%g	the shorter of %f and %e

Demo 4: String Operations

Note: Refer to Module-2 Demo4 file (Sequences-Strings) on LMS for all the steps in detail

edureka!

Sets

Lists

Tuples

Strings

Sets

Dictionaries

Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces { }

Sets can also be created by calling the built-in set function:

```python
x = set('Welcome To Edureka')
print(x)
```

edureka!

When to Use Sets?

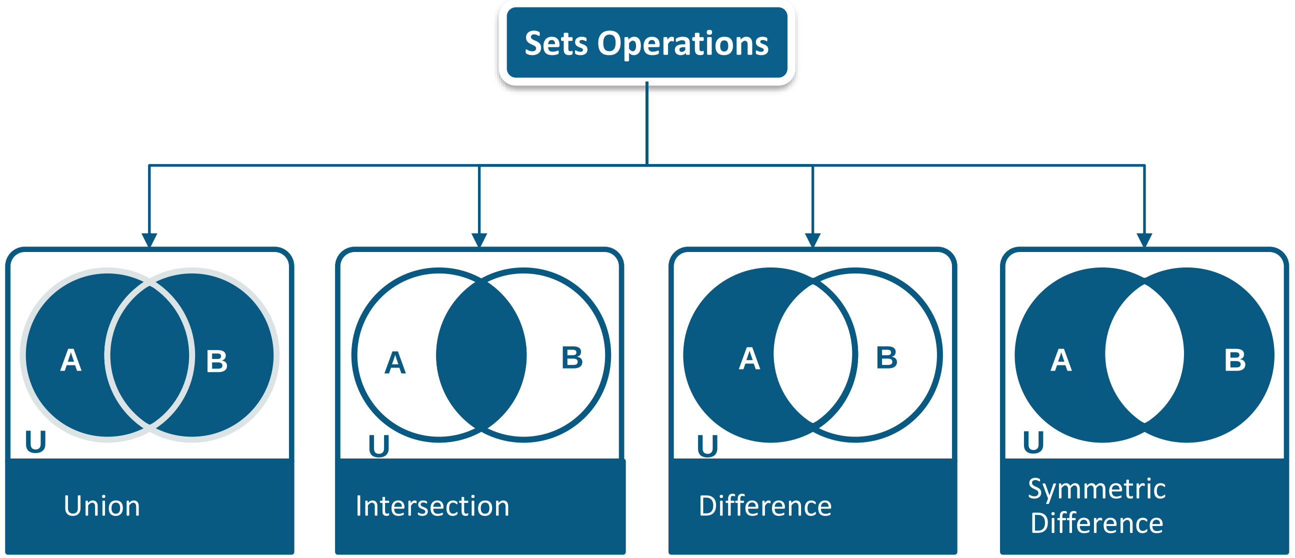
If we wish to collect unique strings or integers from a sequence

When to Use Sets?(Cont.)

College administration is facing problem, because during information feeding, many students are entering the same password and ID

As we know, Sets support unique elements. So, we can convert the lists of IDs and passwords into sets and can get only Unique ones

Sets – Operations



Sets – Operations(Cont.)

Python provides **built-in functions** as well as **operators** for set operations

Set Operation	Operator	Function
Union	|	**union()**
Intersection	&	**intersection()**
Difference	-	**difference()**
Symmetric Difference	^	**symmetric_difference()**

```python
set_a={1,2,4,5,6}
set_b={3,5,6,7,8}
print(set_a|set_b)
print(set_a & set_b)
print(set_a.difference(set_b))
print(set_a.symmetric_difference(set_b))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{5, 6}
{1, 2, 4}
{1, 2, 3, 4, 7, 8}
```

Output

edureka!

Demo 5: Set Operations

Note: Refer to Module-2 Demo5 file (Sequences-Sets) on LMS for all the steps in detail

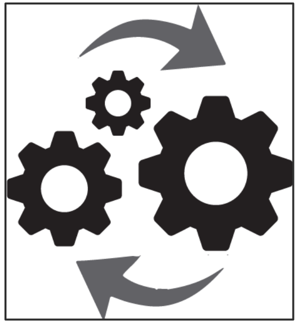
Dictionaries

- Lists
- Tuples
- Strings
- Sets
- **Dictionaries**

Dictionary is an unordered collection of key-value pairs. It is generally used when we have a huge amount of data

Example: dict={1:"Python"}

Operations



- Length
- del d [K]
- Membership Testing

When to Use Dictionaries?

Name	Aadhar Card no.

Annie is a Receptionist in an Office. She has to create records of Employees

She creates an Excel sheet, where she enters employee's names and Aadhar Card no. The Employee's name as a key and Aadhar card no. as value in Dictionary

Dictionaries – Key Points

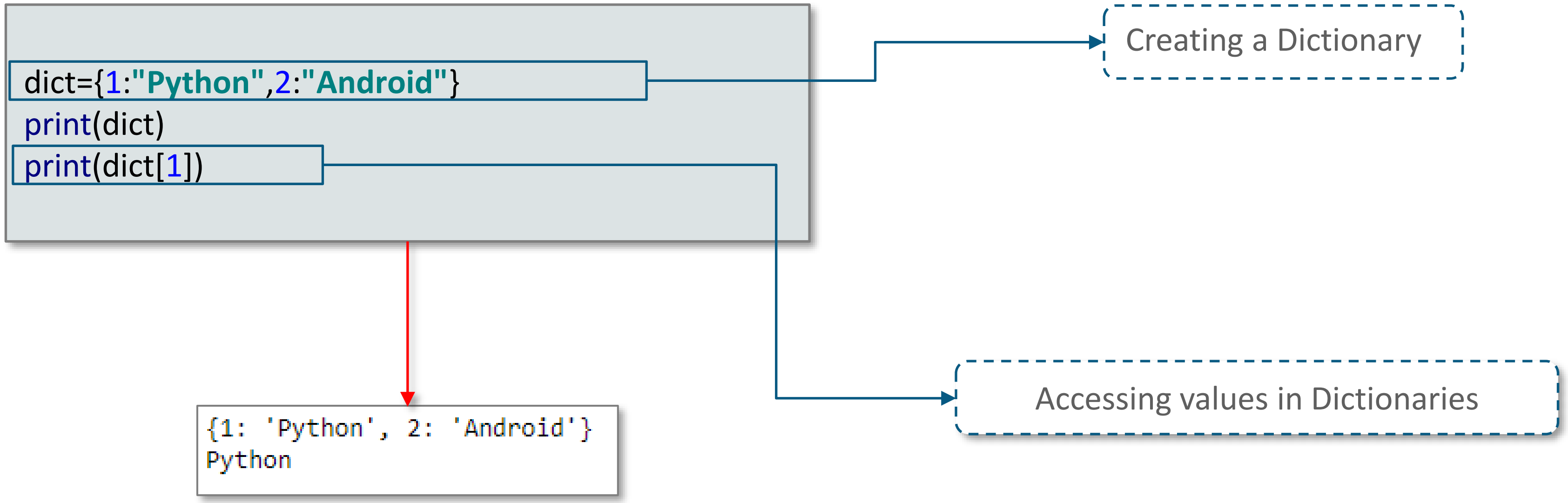
01 Dictionary keys must be unique and immutable

02 Tuple, Number, and String can be dictionary keys because of their immutable nature. Therefore List can’t be a dictionary key

03 Dictionary values can be both mutable and immutable types

04 Deletion of any value in a dictionary will delete the key associated with that value as well

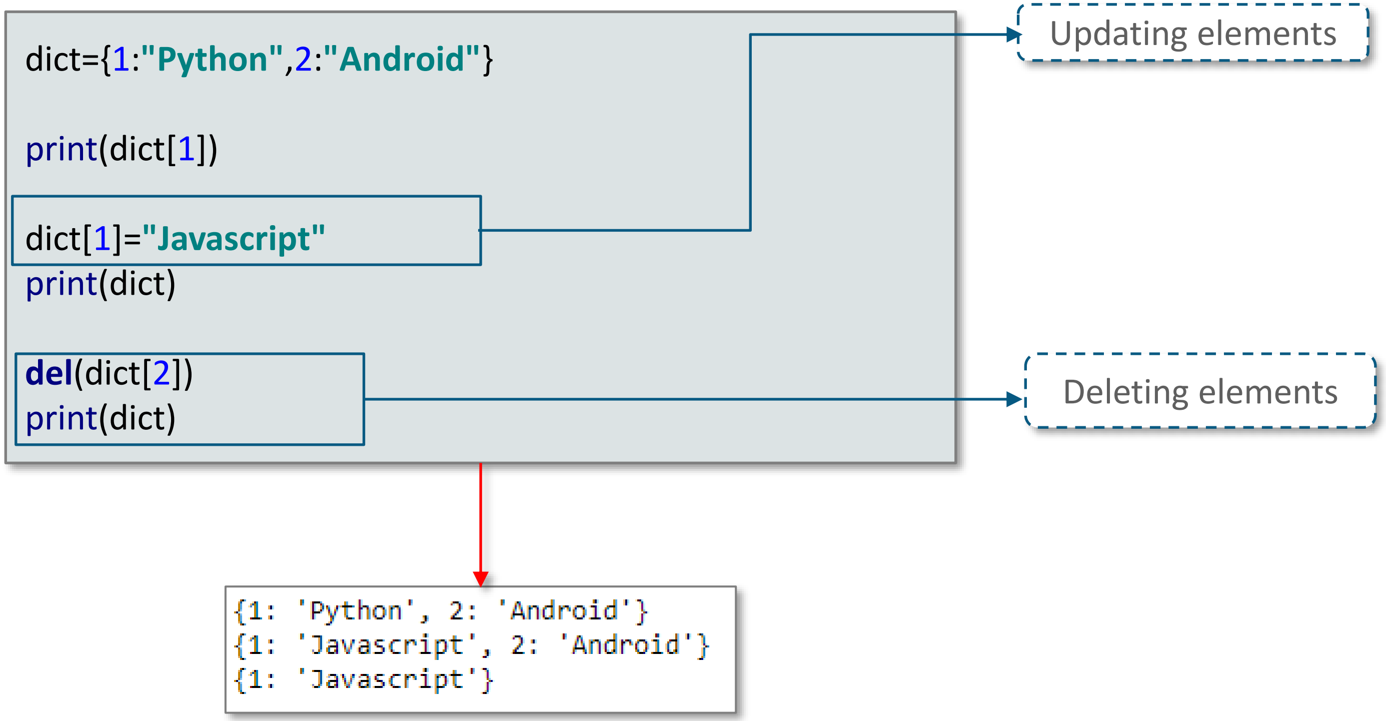
Dictionaries — Example



```
dict={1:"Python",2:"Android"}
print(dict)
print(dict[1])
```

```
{1: 'Python', 2: 'Android'}
Python
```

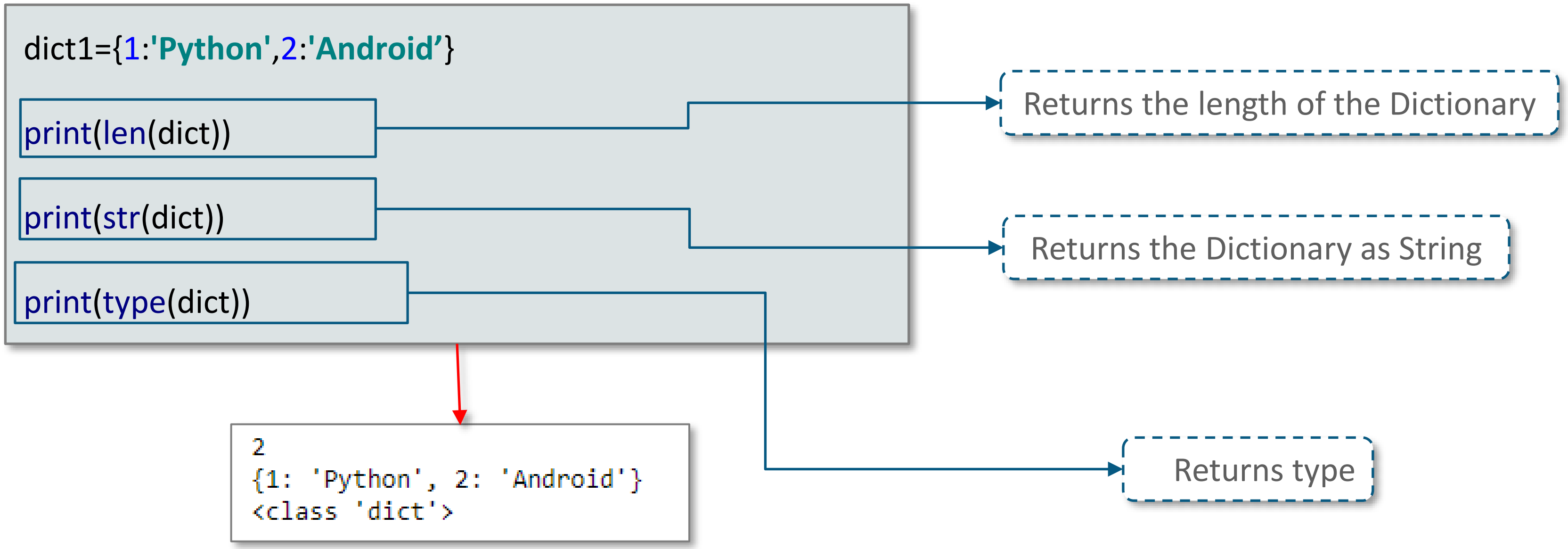
Dictionaries — Updating And Deleting Elements



```
dict={1:"Python",2:"Android"}

print(dict[1])

dict[1]="Javascript"
print(dict)

del(dict[2])
print(dict)
```

```
{1: 'Python', 2: 'Android'}
{1: 'Javascript', 2: 'Android'}
{1: 'Javascript'}
```

Built-in Functions of Dictionaries



```python
dict1={1:'Python',2:'Android’}
print(len(dict))
print(str(dict))
print(type(dict))
```

```
2
{1: 'Python', 2: 'Android'}
<class 'dict'>
```

Returns the length of the Dictionary
Returns the Dictionary as String
Returns type

Built-in Functions of Dictionaries

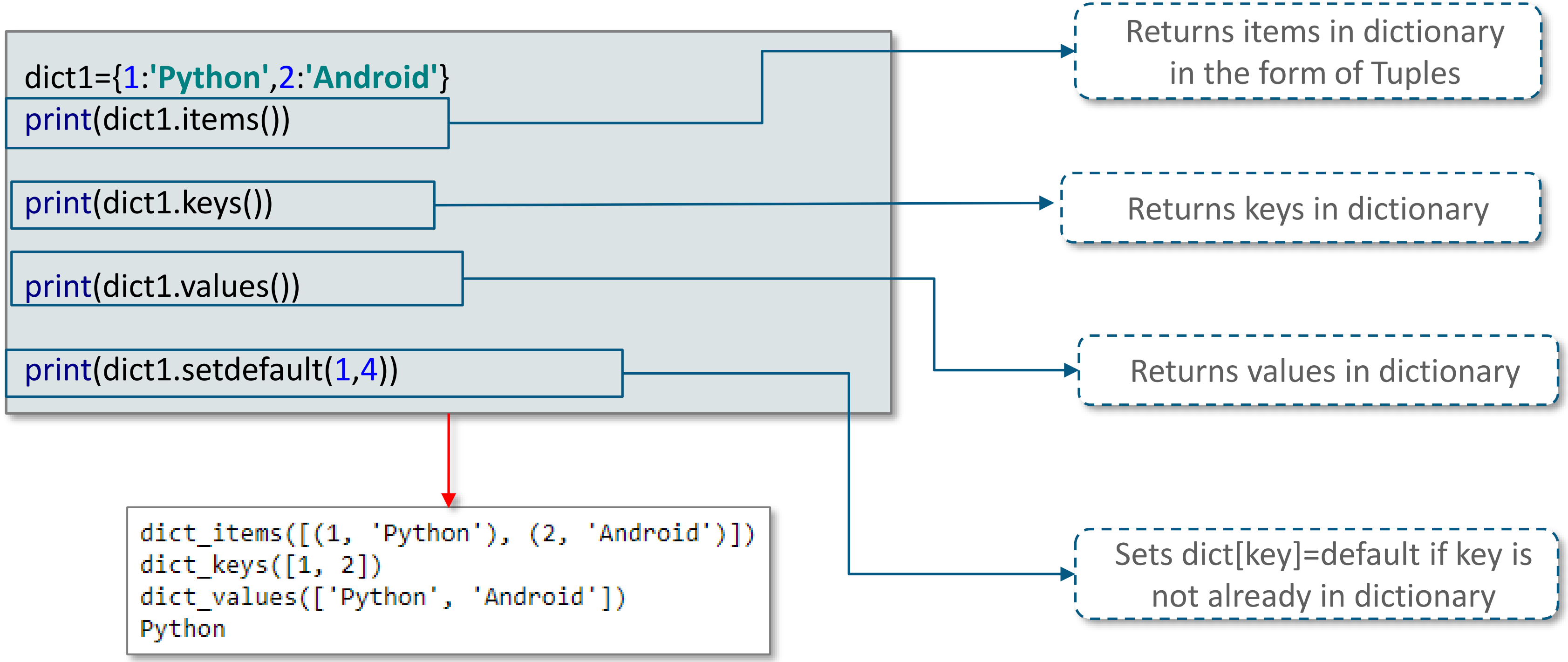
```
rec = {'name': {'first': 'Bob', 'last': 'Smith'},
'jobs': ['dev', 'mgr'],'age': 40.5}
print(rec.get('name'))
```

Returns the value of the key passed

```
{'first': 'Bob', 'last': 'Smith'}
```

edureka!

Methods of Dictionaries



```
dict1={1:'Python',2:'Android'}
print(dict1.items())
print(dict1.keys())
print(dict1.values())
print(dict1.setdefault(1,4))
```

Returns items in dictionary in the form of Tuples

Returns keys in dictionary

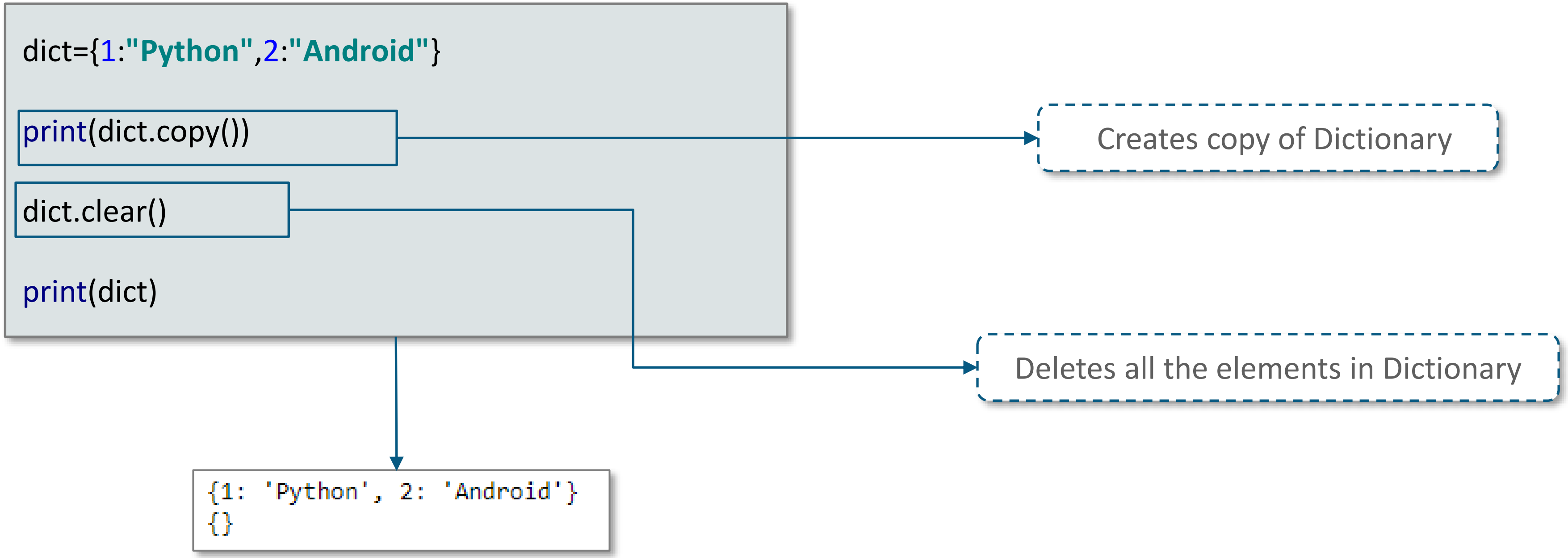
Returns values in dictionary

Sets dict[key]=default if key is not already in dictionary

```
dict_items([(1, 'Python'), (2, 'Android')])
dict_keys([1, 2])
dict_values(['Python', 'Android'])
Python
```

edureka!

Methods of Dictionaries (Cont.)



```
dict={1:"Python",2:"Android"}

print(dict.copy())
dict.clear()

print(dict)
```

Creates copy of Dictionary
Deletes all the elements in Dictionary

```
{1: 'Python', 2: 'Android'}
{}
```

Sorting Keys For Loops

```python
dic={3:'Python',1:'Java',2:'Big Data'}
ks=list(dic.keys())
print(ks)

sk=sorted(ks)
print(sk)

for key in sk:
    print(key,'=>',dic[key])
```

ks consists list of dictionary Keys

sk consists sorted keys of dictionary

Prints sorted keys with their respective values from dictionaries

```
[3, 1, 2]
[1, 2, 3]
1 => Java
2 => Big Data
3 => Python
```

edureka!

Tuple and List In Dictionary

```
#tuple in set
dict={1:(1,2,3),2:(3,4,5)}
print(dict)
print(dict[1][1])


#list in set
dict={1:["Python","Java"],2:[1,3,5,7]}
print(dict)
print(dict[1][0]
```

Tuples are given as elements in Dictionary

Lists are given as elements in Dictionary

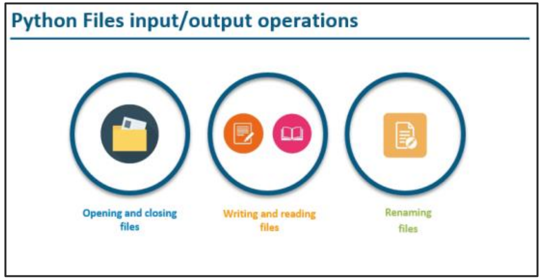
```
{1: (1, 2, 3), 2: (3, 4, 5)}
2
{1: ['Python', 'Java'], 2: [1, 3, 5, 7]}
Python
```

edureka!

Demo 6: Dictionary Operations

Note: Refer to Module-2 Demo6 file (Sequences-Dictionaries) on LMS for all the steps in detail

Summary

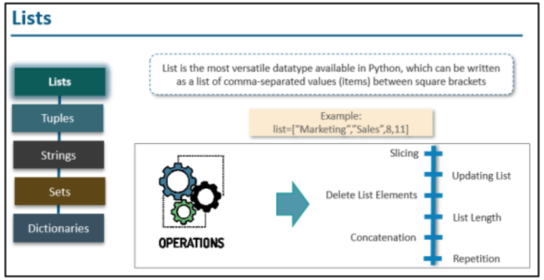


Python Files input/output operations

Opening and closing files

Writing and reading files

Renaming files



Lists

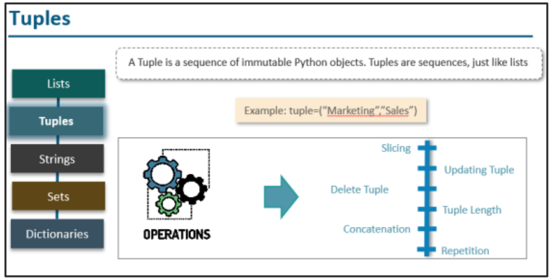
Lists
Tuples
Strings
Sets
Dictionaries

List is the most versatile datatype available in Python, which can be written as a list of comma-separated values (items) between square brackets

Example:
list=["Marketing","Sales",8,11]

OPERATIONS

Slicing
Updating List
Delete List Elements
List Length
Concatenation
Repetition



Tuples

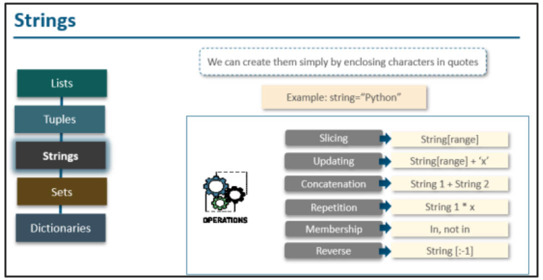
Lists
Tuples
Strings
Sets
Dictionaries

A Tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists

Example: tuple=("Marketing","Sales")

OPERATIONS

Slicing
Updating Tuple
Delete Tuple
Tuple Length
Concatenation
Repetition



Strings

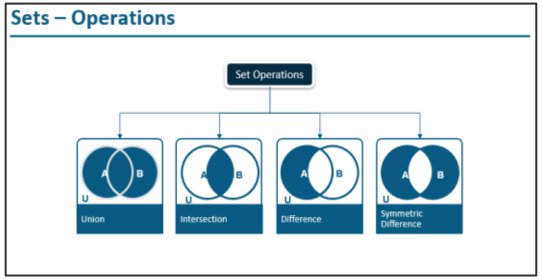
Lists
Tuples
Strings
Sets
Dictionaries

We can create them simply by enclosing characters in quotes

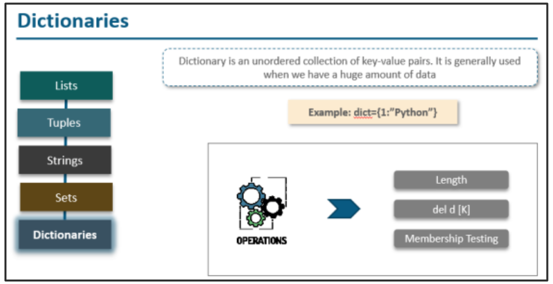
Example: string="Python"

OPERATIONS

Slicing	String[range]
Updating	String[range] + 'x'
Concatenation	String 1 + String 2
Repetition	String 1 * x
Membership	In, not in
Reverse	String [:-1]



Sets – Operations



Dictionaries

Lists
Tuples
Strings
Sets
Dictionaries

Dictionary is an unordered collection of key-value pairs. It is generally used when we have a huge amount of data

Example: dict={1:"Python"}

OPERATIONS

Length
del d [K]
Membership Testing

Questions

edureka!

FEEDBACK

Thank You