python

Programing Language Fundamentals

Done By: Eng. Mohammed Marwan Shahin
Technical Training Specialist
Technology Academy

Training Outlines

- Python Installation
- Python Data type
- Operators
- Control Statements
- Loops
- Functions
- Classes
- Modules

Trainer Profile

Training Area:

- Artificial Intelligence
- Internet of things
- Augmented Reality
- Python Programming (Basics & Advanced)
- Chatbot
- Thingworx Web Application development
- Robotics Process Automations (RPA)

Projects & Achievements

- 1st Rank in Al Mawrid 2019
- AI Certification
- DNA web application
- Smart Parking system
- Humanoid Robot Assistant
- AR mobile Application
- Conducting Training for UAE university instructors and prof.
- Creating a Chatbot for TA

Course Objectives

After completing this course, you will be able to:

- Explain the process to install Python
- Describe the various Python data types
- Discuss the various control statements used in Python
- Explain the various types of loops used in Python
- Explain the process to create and run a function in Python
- Explain the process to create and use a class in Python
- Describe the process to create and import a module in Python

Python Basics

Lesson 01—Introduction to Python

Objectives

After completing this lesson, you will be able to:

- Define Python
- Discuss the history of Python
- Explain how to set up the Python development environment
- Explain how to code Python
- Describe various Python data types

Why Python?

Why do we need Python? Because it:

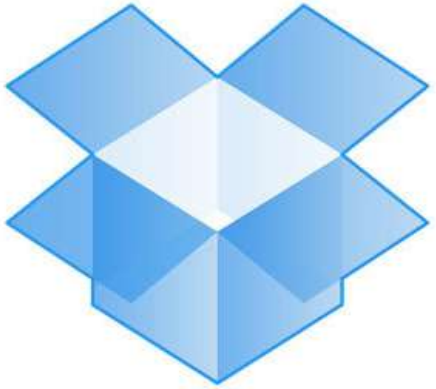
- Is simple and easy to learn
- Has very compact codes
- Has a huge and vibrant community that provides state-of-the-art open source libraries for NLP, data science, and web development
- Is useful for all kind of development needs like web development, scripting, quick prototyping, and data science
- Is handy for fast and agile development
- Is used and promoted by companies like Google, Facebook, and Yahoo

Webs & Apps Build by Python !!

YouTube

Instagram

Google



Dropbox

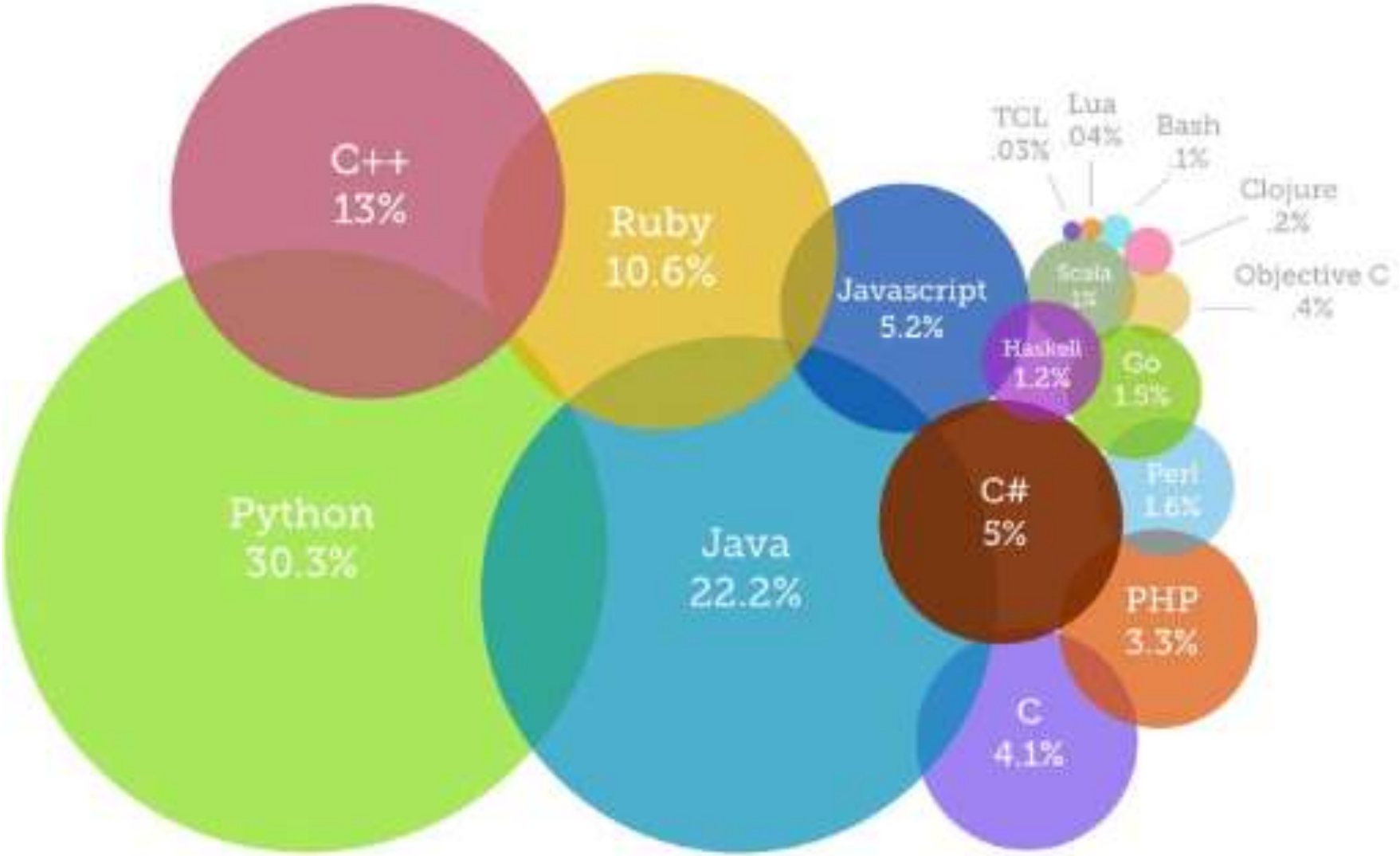
YAHOO!

SurveyMonkey®

Python Applications

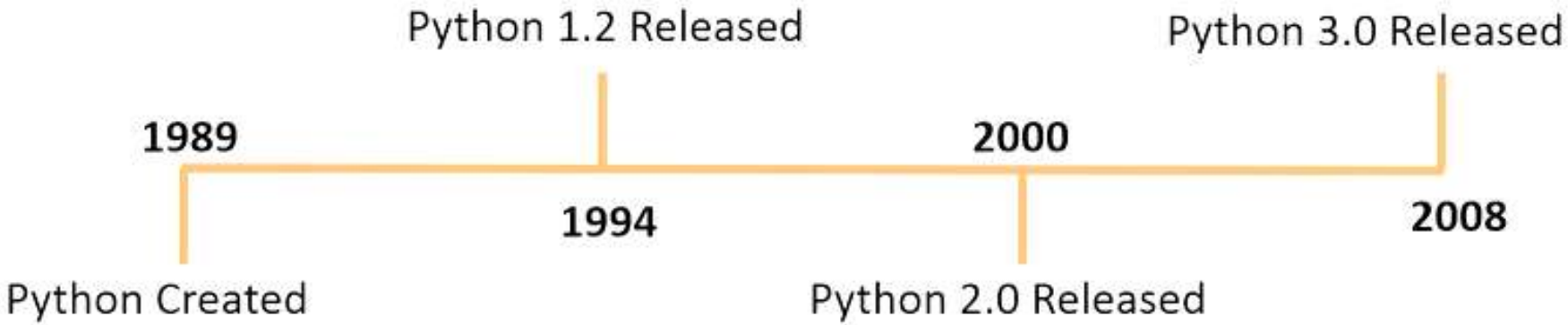
- Web Application Development
- Scientific and Numeric Computing
- GUI Programming
- Business Development

Popularity of Programming Language



Releases

Python has undergone a lot of changes since its origin:



Python 2.7 is the most used version, while Python 3.0 is also widely accepted.

All new libraries in Python primarily use Python 3.0.

Installing Python 3

The steps to install Python on Windows are given below:

1. Download Windows-based installer from https://www.python.org/downloads/release/python-2710/
2. Double click the installer after the download is complete.
3. Follow the instructions and finish the installation.

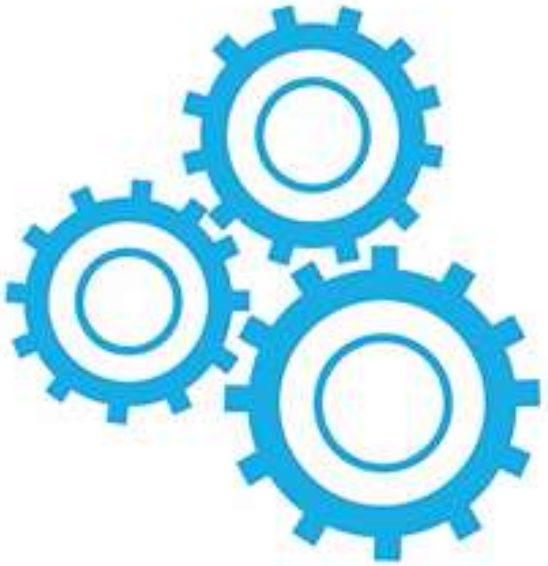
! While clicking the "Next" button, it is recommended to read the agreement properly.

Writing Comments in Python

How can you write comments in Python?

For bigger and complex program:

- Add # before a line to make it a line comment.
  Example:
  #This is a single line comment
- Add multiline comments by enclosing text in """.
  Example:
  """ This is a multiline comment. It can span across multiple lines."""



Hello Word

```
>>> Print(“Hello World”)
```

Try & check the output of the followings:

```
>>> print("Hello\nWorld!")

>>> # print('Hello World!')
```

Python Basics

Lesson 02—Python Data Types

Objectives

After completing this lesson, you will be able to:

- Describe the types of variables in Python and their uses
- Define operators and operands
- Explain the relationship between operands and variables
- Explain how arithmetic operations are used on numerical values

Variable

- Preserved memory space, can hold any value and it’s assigned to a term.

Example : `>>> age = 4`

Variable **Value**

Keyword

- Python keyword is a unique programming term intended to perform some action.
- There are 33 keywords in python.
- They represent the syntax and structure of a python program.
- Since all of them are reserved , so you can’t use their names for defining variables, classes or functions.
- All keywords in python are case sensitive.

Keyword

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

Variable

- A variable is a named memory space to store values.
- Based on the data type of variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

Variable Assignment

Syntax:

Variable_name=value

Multiple Assignment

Syntax:

Variable_name1, Variable_name2, Variable_name3,…= value1,value2,value3,…

Variable

A variable is an entity that can hold a value.

Features:

- A variable should have a value.
- An assignment statement creates new variables and gives them values.

Example:

>>>age = 25

Multiple Assignment

```
>>> person1, person2, person3= "mohammed" , "Marwan", "shahin“

>>> var1 = var2 = var3 = “Mohammed”
```

Types of Variables

The six basic types of variables in Python are:

- String
- Integer
- Float
- Boolean
- List
- Dictionary

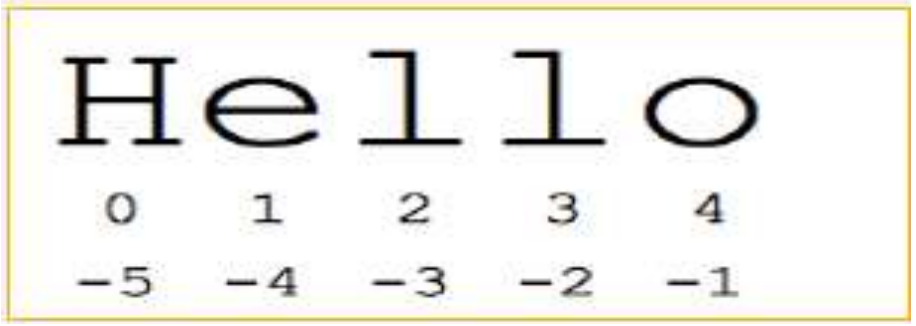
Example:

>>>type("Hello world")

Output: <type 'str' >

Types of Variables – String

- String are one or more characters enclosed in single quotes or double quotes.
- Python supports multi-line strings which require a triple quotation mark at the start and at the end.
- String are immutable in python, memory allocated once and reused after. They cant be altered.



Types of Variables – String

- We can obtain multiple characters from a string using slicing, we can obtain the 0 to 4th and 8th to the 12th element:

- We can also input a stride value as follows, with the '2' indicating that we are selecting every second variable:

Note:

- We can also incorporate slicing with the stride. In this case, we select the first five elements and then use the stride:

```
>>>a= "hello word"
>>>print(a[0:5:2])
hlo
```

Types of Variables – String

A string is used to store textual information, such as a name or an address.

Example:

```
>>>client_first_name= "John D"
>>>print(client_first_name)
```

Concatenate strings using + sign:

```
>>>client_name = client_first_name + "Rockerfeller"
client_name is new variable with value
        "John D Rockerfeller"
```

Data Types (Strings)

```
>>> firstname = "mohammed"
>>> secondname = "marwan"
>>> fullname = firstname + secondname
>>> fullname
'mohammedmarwan'
>>> fullname = firstname + " " + secondname
>>> fullname
'mohammed marwan'
>>> fullname [4]
'm'
>>> fullname [0:7]
'mohamme'
>>> fullname [0:8]
'mohammed'
>>> fullname[0:]
'mohammed marwan'
```

Notes:

- **Back slash "n" represents a new line**

```
>>>print(" Mohammed Marwan \n is the best" )
```

- **Similarly, back slash "t" represents a tab:**

```
>>>print(" Mohammed Marwan \t is the best" )
```

- **If you want to place a back slash in your string, use a double back slash:**

```
>>>print(" Mohammed Marwan \\ is the best" )
```

Len

We can find the number of characters in a string by using 'len', short for length:

```
>>>len(“Mohammed Marwan")
```

String Operation (upper)

There are many string operation methods in Python that can be used to manipulate the data. We are going to use some basic string operations on the data.

Let's try with the method "upper"; this method converts upper case characters to lower case characters:

```
>>>A="Thriller is the sixth studio album"
>>>print("before upper:",A)
>>>B=A.upper()
>>>print("After upper:",B)
```

String Operation (replace ,find)

The method replaces a segment of the string, i.e. a substring with a new string. We input the part of the string we would like to change. The second argument is what we would like to exchange the segment with, and the result is a new string with the segment changed:

```
A=“Mohammed Marwan is the best"
B=A.replace(' Mohammed ', 'Janet')
B
```

The method "find" finds a sub-string. The argument is the substring you would like to find, and the output is the first index of the sequence. We can find the sub-string “amm" or “ed".

```
Name=" Mohammed Marwan"
Name.find(‘ed')
```

Placeholders

It’s like a template where you can insert values.

```
>>> sen = "Hello %s, you're invited to my birthday"
>>> print(sen%("Avi"))
Hello Avi, you're invited to my birthday
>>> arr = ["Bob", "Jake", "Michelle"]
>>> for i in arr:
        print(sen%(i))


Hello Bob, you're invited to my birthday
Hello Jake, you're invited to my birthday
Hello Michelle, you're invited to my birthday
>>> sen = "Hello %s %s, you're invited to my birthday"
>>> sen%("Barack", "Obama")
"Hello Barack Obama, you're invited to my birthday"
>>> sen = "I am %s and my age is %d"
>>> sen % ("Avi", 14)
'I am Avi and my age is 14'
>>>
```

Types of Variables – Numeric Type

Integer and Float are two variables types that are categorized as numeric types.

Integer

Stores real numbers

Example:

```
>>> age = 27
```

or

```
>>> age = int("27")
```

Float

Stores decimals values

Example:

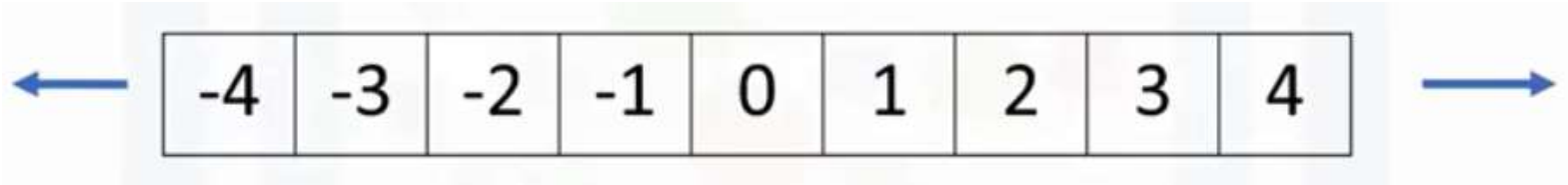
```
>>> pi_value = 3.14
```

or

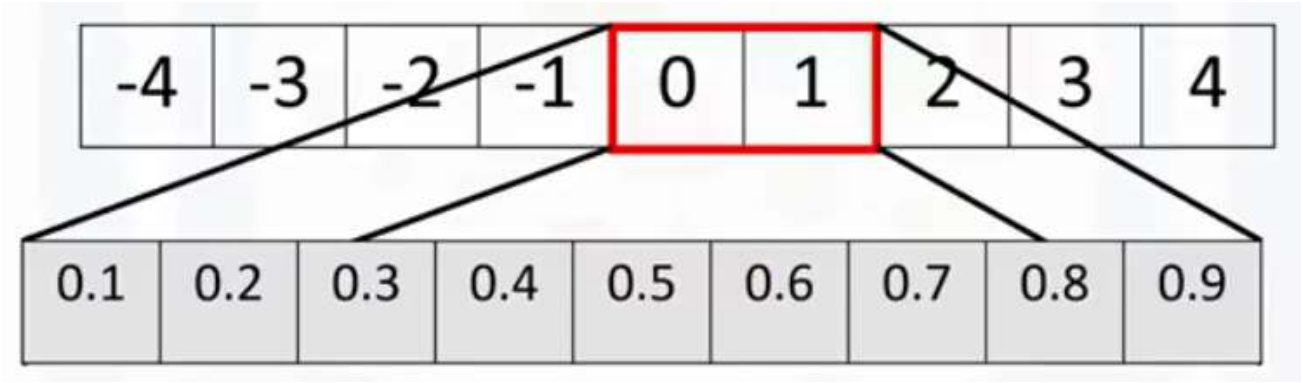
```
>>> pi_value = float("3.14")
```

Types of Variables – Numeric Type

Int:



Float:



Types of Variables – Boolean Variables

Boolean variables store True or False values.

Example:

```
>>>is_python=True
>>>is_python
Output : True
```

Types of Variables – Boolean Variables

- Boolean can have two values – True or False
- These values are constants
- Used to assign or compare Boolean values
- An expression can also produce Boolean values
- True is 1 and False is 0

For Example:

If a = 12 and b =10

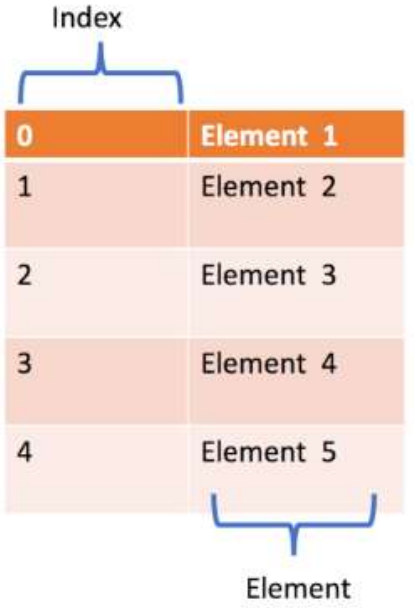
Print a>b	Return True
Print a==10	Return False

Types of Variables – List

- Python lists are the data structure that is capable of holding different type of data.
- It is a container that holds other objects in a given order. Different operation like insertion and deletion can be performed on lists.
- A python list is enclosed between square([]) brackets.
- The elements are stored in the index basis with starting index as 0.
- Python lists are mutable ,i.e., Python will not create a new list if we modify an element in the list.

Types of Variables – List

A list is a sequenced collection of different objects such as integers, strings, and other lists as well. The address of each element within a list is called an 'index'. An index is used to access and refer to items within a list.

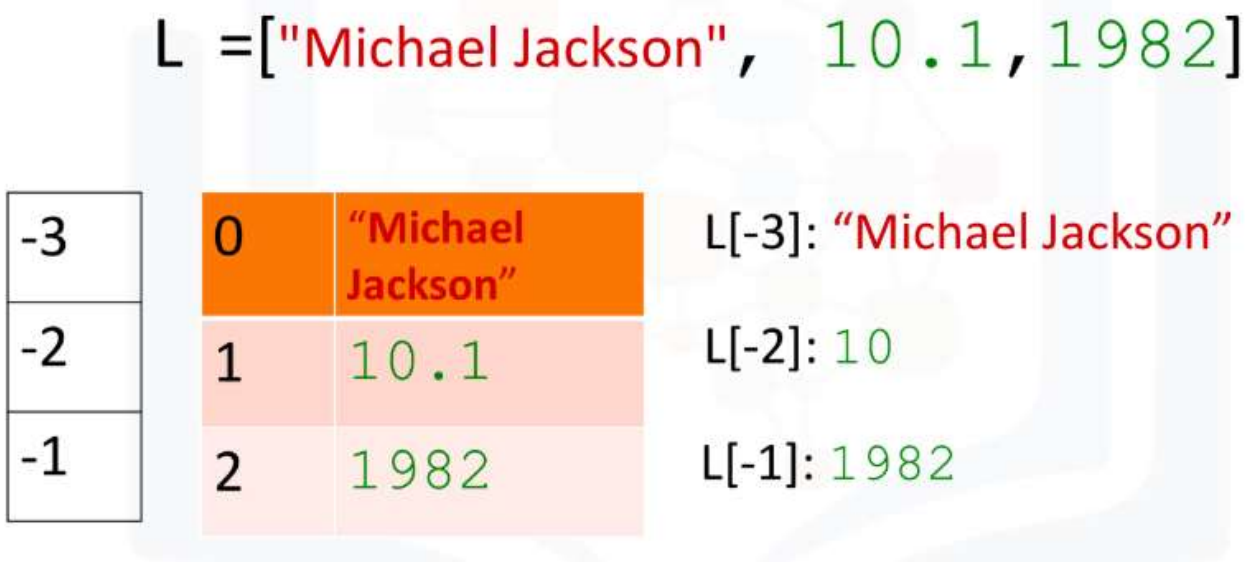


[Element 1 , Element 2 , Element 3 , Element 4, Element 5]

Index 0 1 2 3 4

Types of Variables – List

We can use negative and regular indexing with a list :




```
>>>print('the same element using negative and positive indexing:\n Postive:',L[0],'\n Negative:' , L[-3]  )
>>>print('the same element using negative and positive indexing:\n Postive:',L[1],'\n Negative:' , L[-2]  )
>>>print('the same element using negative and positive indexing:\n Postive:',L[2],'\n Negative:' , L[-1]  )
```

Adding Elements to a List

You can add more elements to a list using the extend() and append() functions.

Using extend()

Example:

```
>>>
country.extend(["England",
"China"])
>>> print(country)
output: ["India", "United
States", "England", "China"]
```

Using append() for individual elements

Example:

```
>>>country.append("England
")
>>> country.append("China")
>>> print country
output: ["India", "United
States", "England", "China"]
```

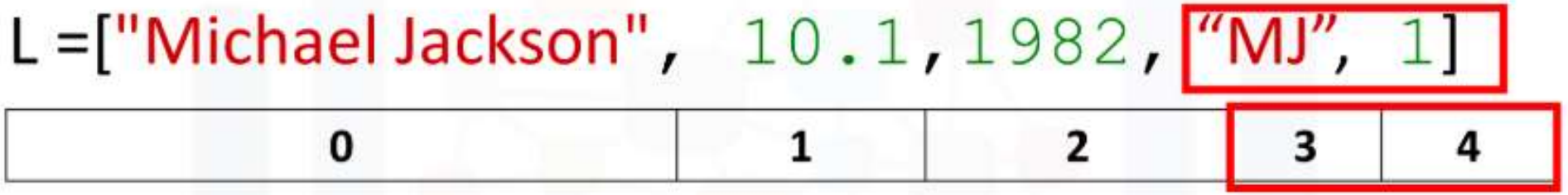
Using append() for a nested list

Example:

```
>>>country.append(
['England', 'China'])
>>>print(country)
output: ["India", "United
States", ["England", "China"]]
```

Types of Variables – List

We can also perform slicing in lists. For example, if we want the last two elements, we use the following command:




```
>>> L[3:5]
['MJ',1]
```

Accessing The Elements of a List

You can access a single element or multiple elements of a list using an index.

Accessing a single element

Example:

>>>country=["India", "United States", "England", "China"]
>>>country[0]
output: "India"

Accessing multiple elements

Example:

>>>country[1:3]
output: ["United States", "England"]
>>>country[:2]
output: ["India", "United States"]
>>> country[3:]
output: ["China"]

List methods

As lists are mutable, we can change them. For example, we can change the first element as follows:

```
>>>A=[“mohammed",10,1.2]
>>>print('Before change:', A)
Before change: [‘mohammed', 10, 1.2]

>>>A[0]=ahmad'
>>>print('After change:', A)
After change: [ahmad', 10, 1.2]
```

List methods (Split)

We can convert a string to a list using 'split'. For example, the method **split** translates every group of characters separated by a space into an element in a list:

```
>>>mohammed marwan'.split()
[mohammed', marwan']
```

We can use the split function to separate strings on a specific character. We pass the character we would like to split on into the argument, which in this case is a comma. The result is a list, and each element corresponds to a set of characters that have been separated by a comma:

```
>>>'A,B,C,D'.split(',')
['A', 'B', 'C', 'D']
```

List methods

- employee = ['mohammed', ' ali' , ' ahmad']
- a= employee [0:2]
- print (a)
- employee[0]= 'sultan'
- print(employee)
- employee.append ('mohammed')
- print(employee)
- employee.append (['abas','eslam'])
- print (employee)
- employee.extend (['saeed','adel'])
- print(employee)
- print(employee.insert(0,'mm'))
- print(employee)
- employee.reverse()
- print(employee)

Lists

We can create a list so that we can use indexing to pick the right value from the list.

```
>>> list1=["mohammed","Marwan"]
>>> list1(1)
Traceback (most recent call last):
  File "<pyshell#40>", line 1, in <module>
    list1(1)
TypeError: 'list' object is not callable
>>> list1[1]
'Marwan'
>>> list[1] = "Shahin"
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    list[1] = "Shahin"
TypeError: 'type' object does not support item assignment
>>> list1["shahin"]
Traceback (most recent call last):
  File "<pyshell#43>", line 1, in <module>
    list1["shahin"]
TypeError: list indices must be integers, not str
>>> list1[1]="Shahin"
>>> list1
['mohammed', 'Shahin']
>>> list1[1]=" "
>>> list1
['mohammed', ' ']
>>> del list1[1]
>>> list1
['mohammed']
```

List Function

- Del
- Min
- Max
- Len
- Append (ADD)
- Count

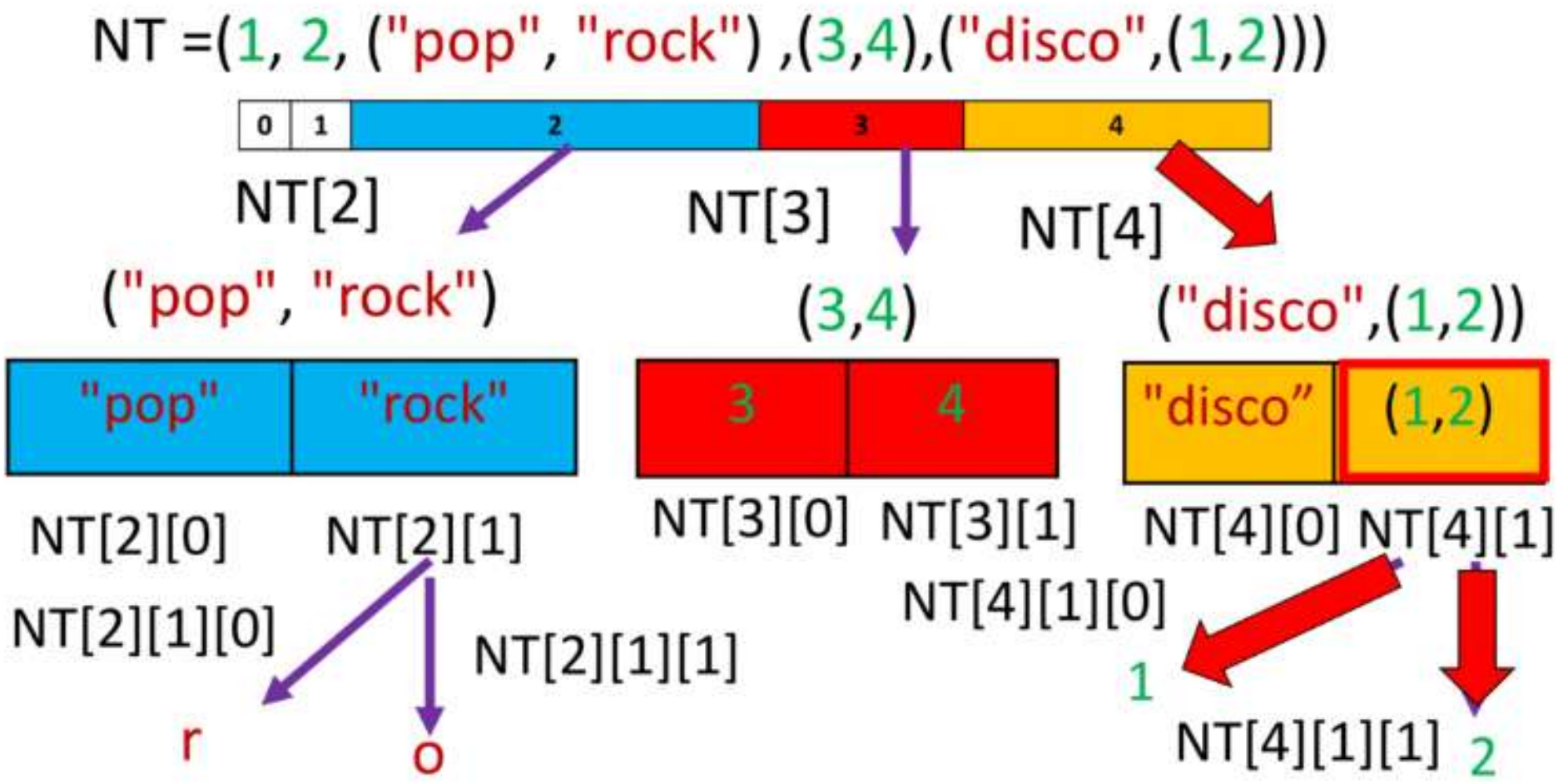
```
>>> shoppingList2[2]  = "chocolate"
>>> shoppingList2
['eggs', 'carrots', 'chocolate', 'cherries', 'apples']
>>> del shoppingList2[4]

SyntaxError: unexpected indent
>>> del shoppingList2[4]
>>> shoppingList2
['eggs', 'carrots', 'chocolate', 'cherries']
>>> array1= [23, 54, 64]
>>> array2 = [43, 23]
>>>
>>> array3 =  array1 + array2
>>> array3
[23, 54, 64, 43, 23]
>>> len(shoppingList2)
4
>>> numArray = [36, 75, 10004, 1, -5]
>>> max(numArray)
10004
>>> min(numArray)
-5
>>> shoppingList2.append("Brocolli")
>>> shoppingList2
['eggs', 'carrots', 'chocolate', 'cherries', 'Brocolli']
>>> shoppingList2.count("chocolate")
1
```

Type of Variables - Tuples

- A tuple is a sequence of immutable python objects, therefore tuple cannot be changed.
- The objects are enclosed within parenthesis and separated by comma.
- Tuple is similar to list. Only the difference is that list is enclosed between square bracket, tuple between parenthesis ()and list have mutable objects whereas Tuple have immutable objects.

Type of Variables - Tuples



Tuples

- Cant (change , add or delete)
- Index is possible

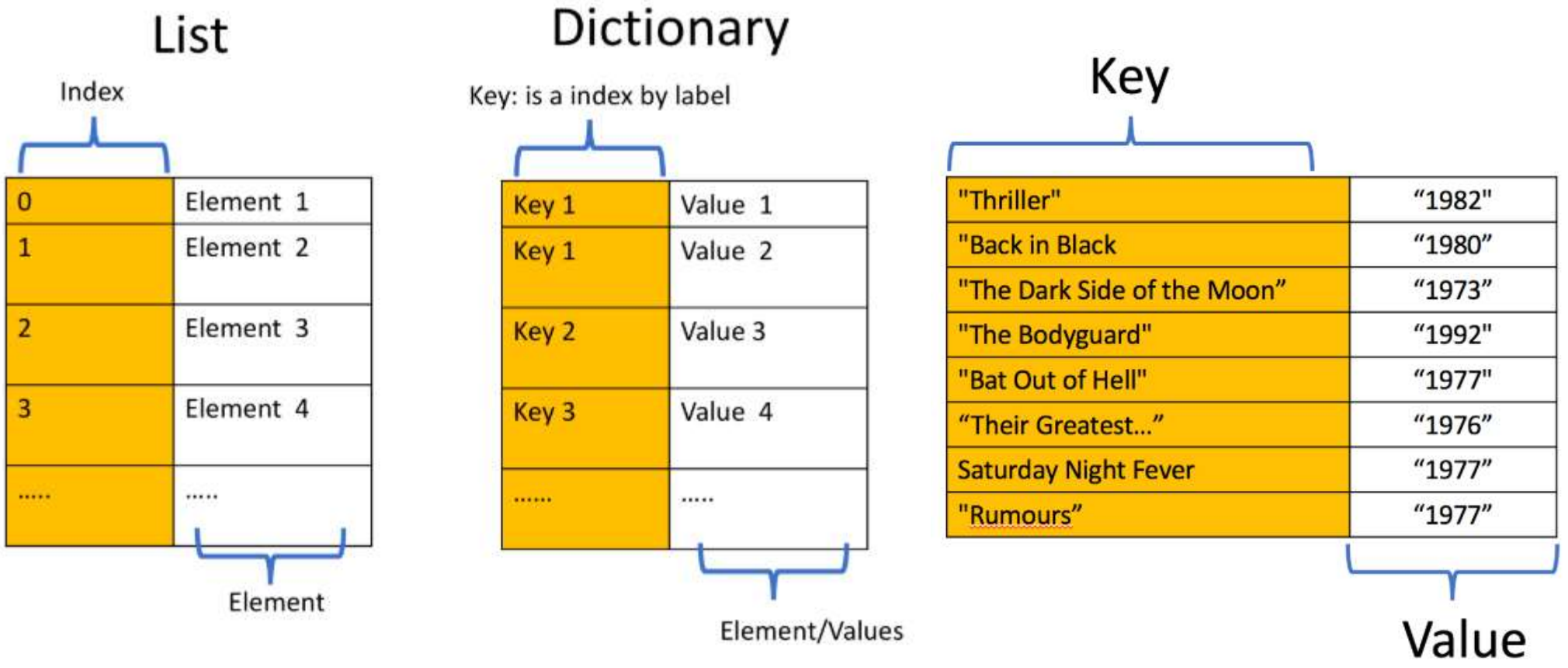
```
>>> tup1 = ("Maths", 23, "Dogs")
>>> tup1.append("asd")
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    tup1.append("asd")
AttributeError: 'tuple' object has no attribute 'append'
>>> del tup1["Maths"]
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    del tup1["Maths"]
TypeError: 'tuple' object does not support item deletion
>>> tup1 = ("Cheese")
>>> tup1
'Cheese'
>>> tup1 = ("Maths", 23, "Dogs")
>>> tup1[0]
'Maths'
>>> tup1[1]
23
>>> tup1[0:2]
('Maths', 23)
>>> tup1[0:3]
('Maths', 23, 'Dogs')
>>> del tup1
>>> tup1 = ("Maths", 23, "Dogs")
```

Type of Variables - Dictionaries

- Dictionary is an unordered set of key and value pair, enclosed within curly braces {}.
- The pair i.e., key and value is known as item.
- The key passed in the item must be unique.
- The key and value is separated by a colon(:).
- Items are separated from each other by a comma(,).

Type of Variables - Dictionaries

A dictionary consists of keys and values. It is helpful to compare a dictionary to a list. Instead of the numerical indexes such as a list, dictionaries have keys. These keys are labels that are used to access values within a dictionary.



Type of Variables - Dictionary

Similar to a hash map, a dictionary is used to store anything as a key, value pair.

An example of a dictionary:

>>>phone_dict = {}

In summary, like a list, a dictionary holds a sequence of elements. Each element is represented by a key and its corresponding value. Dictionaries are created with two curly braces containing keys and values separated by a colon. For every key, there can only be a single value, however, multiple keys can hold the same value. **Keys can only be strings, numbers, or tuples, but values can be any data type.**

Adding Elements to a Dictionary

You can add individual elements and multiple elements to a dictionary using the assignment operator and the update() function, respectively.

Adding Single Elements

Example:

```
>>>phone_dict['john'] =
9876543210
>>>phone_dict['harry']
= 9786543210
```

Adding Multiple Elements

Example:

```
>>>phone_dict.update(
{'andy' : 9876565612 ,
'jane' : 9876123450})
```

Any immutable object can be a key; **Example**: *>>>phone_dict[('sandy' , 'ammy')] = 9812345670*
>>>phone_dict['rose'] = [9123456780,9871234567]

Accessing The Elements Of A Dictionary

You can access the elements of a dictionary using key names.

Example:

>>>phone_dict['john']

output: 9876543210

Dictionary Methods

Different methods used with dictionaries are explained below:

Get()

Returns the value associated with the key passed in the method

Example:

>>>phone_dict.get('andy')

output: 9876565612

Keys()

Returns all keys of a dictionary

Example:

>>>phone_dict.keys()

output: ['john' , 'harry' , 'andy', ['jane' , 'sandy'], 'ammy' , 'rose']

Dictionary Methods

Some more methods are:

Values()

Prints all values of a dictionary

Example:

```
>>>phone_dict.values()
output:
[9876543210,9786543210,9
876565612,9876123450,
9812345670,
[9123456780,9871234567]]
```

Items()

Returns all elements of a dictionary

Example:

```
>>>phone_dict.items()
output: ['john' = 9876543210 ,
'harry' =9786543210,
'andy' = 9876565612 , 'jane' =
9876123450 ,
['sandy','ammy']=9812345670 ,
'rose' =
[9123456780,9871234567] ]
```

Dictionaries

- Like a map key , using {} brackets, it’s case sensitive

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  5 2014, 20:15:28)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> students = ["Eric", 14, "Bob", 12, "Tina", 26, "Chris", 15]
>>> students = {"Eric":14, "Bob": 12, "Tina": 26}
>>> students["Bob"]
12
>>> students["Bob"] = 13
>>> students["Bob"]
13
>>> del students["Bob"]
>>> students
{'Eric': 14, 'Tina': 26}
>>> students["eric"]
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    students["eric"]
KeyError: 'eric'
>>>
```

Dictionary Functions

- Clear
- Del
- Len
- Getting only Keys or Values
- update

```
>>> students = {"Jobs": 15, "Marc": 24, "Avi": 14}
>>> students.clear()
>>> students
{}
>>> del students
>>> students
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    students
NameError: name 'students' is not defined
>>> students = {"Jobs": 15, "Marc": 24, "Avi": 14}
>>> len(students)
3
>>> students.keys
<built-in method keys of dict object at 0x487d8ac>
>>> students.keys()
dict_keys(['Marc', 'Jobs', 'Avi'])
>>> student.values()
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    student.values()
NameError: name 'student' is not defined
>>> students.values()
dict_values([24, 15, 14])
>>> students2 = {"Job": 15, "Mar": 24, "Av": 14}
>>> students.update(students2)
>>> students
{'Mar': 24, 'Marc': 24, 'Av': 14, 'Job': 15, 'Jobs': 15, 'Avi': 14}
```

Exercise

List:

Create a list 'a_list' , with the following elements 1, “hello”, [1,2,3] and True.
Find the value stored at index 1 of 'a_list'.
Retrieve the elements stored at index 1,2, and 3 of 'a_list'.
Concatenate the following lists A=[1,'a'] abd B=[2,1,'d']:

Dictionary:

The Albums 'Back in Black', 'The Bodyguard' and 'Thriller' have the following music recording sales in millions 50, 50 and 65 respectively:

a) Create a dictionary “album_sales_dict” where the keys are the album name and the sales in millions are the values.
b) Use the dictionary to find the total sales of "Thriller":
c) Find the names of the albums from the dictionary using the method "keys":
d) Find the names of the recording sales from the dictionary using the method "values":