CA326: Advanced Embedded Programming
Chapter 4 and Chapter 5
(Raspberry PI and Python Programming)

Raspberry PI

Write down features and specifications of Raspberry Pi 4 Model B.

- It can be connected with CSI camera module.
- Up to 100 Mbps Ethernet speed.
- Bluetooth 5.0 connectivity.
- Dual band wifi at 2.4 GHz and 5 GHz frequency
- External MicroSD card is supported.
- 2 microSD card.
- 2 USB3.0
- 2 USB 2.0
- 1.5 GHz CPU speed
- Light wright

Where Raspberry Pi can be used?? (What are the uses of Raspberry PI?)

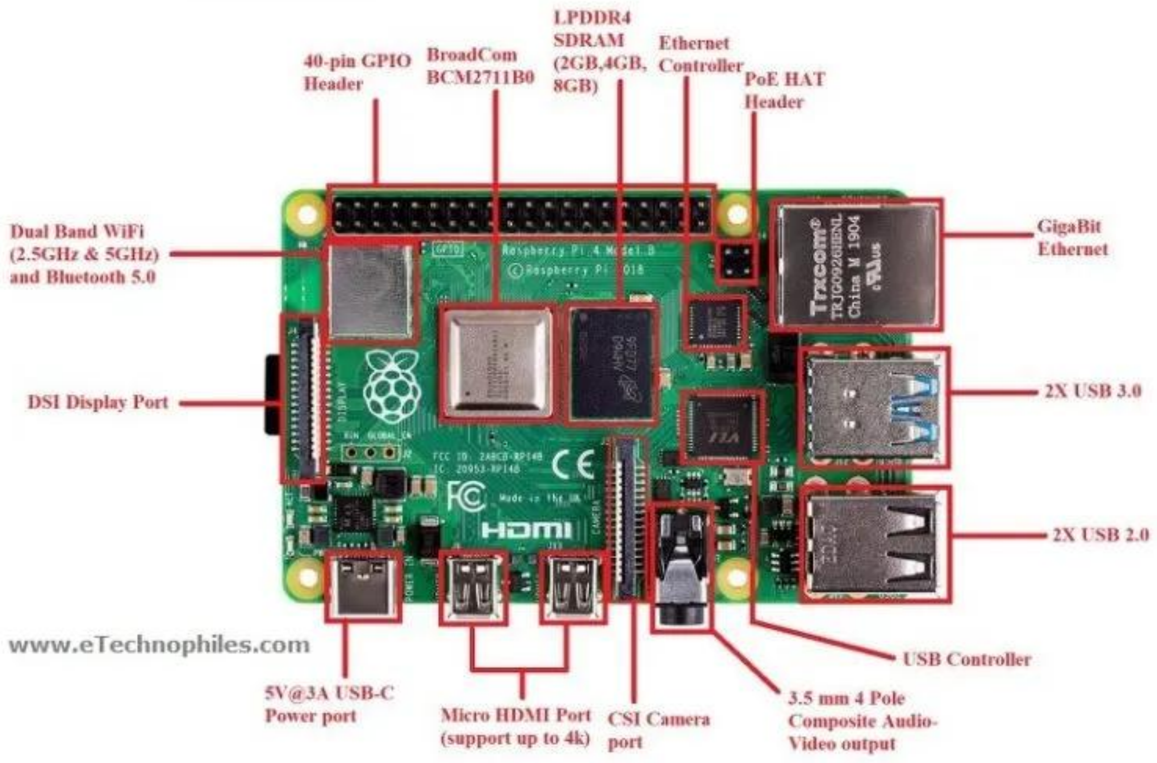
- Tinker (Create) the circuits.
- Playing games
- Write codes / programs
- Create own software.
- Making gaming devices
- Used in fitness tools
- Create spreadsheet
- Create documents
- Editing the photos.
- Surfing the internet
- Various Network activities

List and Explain various Input / output components / devices which can be used with Raspberry PI Projects or applications.

1. Monitor
2. TV screen
3. USB Hub
4. Keyboard

5. Mouse
6. SD card
7. Micro SD card
8. Wifi adaptor
9. External hard drive
10. Speaker
11. Camera Module
12. Power supply
13. Cables
    a. HDMI cable
    b. Ethernet cable
    c. Audio cable
14. Sensors
15. LEDs and Switches

Explain the schematic of Raspberry PI 4 model B with detailed PINs.



Raspberry PI can have Modules as shown in above figure.
Various modules are

- CSI camera port
- 5v / 3 amp USB-C power port

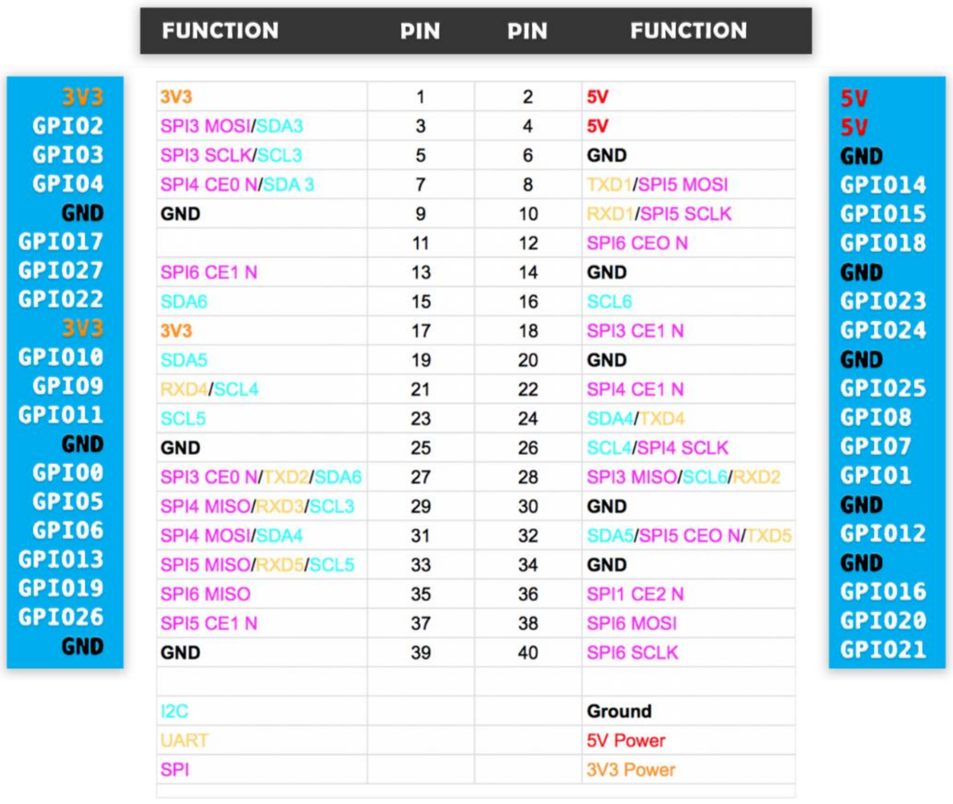
- Two Micro HDMI port
- Composite audio video output
- USB controller
- Two USB 3.0
- Giga bit Ethernet
- Dual band wifi
- 40 Pin GPIO

Total Pins are 40

- 28 GPIO Pins
- 8 Ground Pins
- 4 power pins (5v , 3.3 v)

It is having Pins for UART, SPI communication and I2C communications.

Explain the functional or PIN diagram or Raspberry PI with details of each Pins.



	FUNCTION	PIN	PIN	FUNCTION	
3V3	3V3	1	2	5V	5V
GPIO2	SPI3 MOSI/SDA3	3	4	5V	5V
GPIO3	SPI3 SCLK/SCL3	5	6	GND	GND
GPIO4	SPI4 CE0 N/SDA 3	7	8	TXD1/SPI5 MOSI	GPIO14
GND	GND	9	10	RXD1/SPI5 SCLK	GPIO15
GPIO17		11	12	SPI6 CEO N	GPIO18
GPIO27	SPI6 CE1 N	13	14	GND	GND
GPIO22	SDA6	15	16	SCL6	GPIO23
3V3	3V3	17	18	SPI3 CE1 N	GPIO24
GPIO10	SDA5	19	20	GND	GND
GPIO9	RXD4/SCL4	21	22	SPI4 CE1 N	GPIO25
GPIO11	SCL5	23	24	SDA4/TXD4	GPIO8
GND	GND	25	26	SCL4/SPI4 SCLK	GPIO7
GPIO0	SPI3 CE0 N/TXD2/SDA6	27	28	SPI3 MISO/SCL6/RXD2	GPIO1
GPIO5	SPI4 MISO/RXD3/SCL3	29	30	GND	GND
GPIO6	SPI4 MOSI/SDA4	31	32	SDA5/SPI5 CEO N/TXD5	GPIO12
GPIO13	SPI5 MISO/RXD5/SCL5	33	34	GND	GND
GPIO19	SPI6 MISO	35	36	SPI1 CE2 N	GPIO16
GPIO26	SPI5 CE1 N	37	38	SPI6 MOSI	GPIO20
GND	GND	39	40	SPI6 SCLK	GPIO21
	I2C			Ground	
	UART			5V Power	
	SPI			3V3 Power	

- There are total 40 pins.
- 28 pins are GPIO pins.
- 12 pins are Power pins.

Power Pins of Raspberry PI:

- Total **ground pins are pin 6, 9, 14, 20, 25, 30, 34, 39**.
- Total **5V pins are Pin 2, 4.**
- Total **3V Pins are Pin 1 , 17**.
- All ground pins are connected internally and they are used to provide ground voltage or zero voltage to devices.

GPIO Pins (Digital Input / Output Pins):

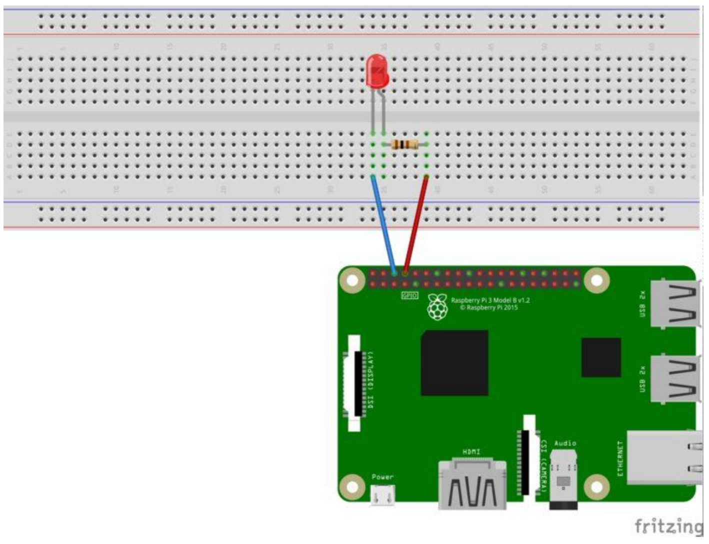
- There are total 28 GPIO pins.
- Almost every device need GPIO connection for communicate.
- These 28 pins can be used either as an input or an output.
- All GPIO pins are used not only for inputs and outputs but they can be used for other specific / special functions also.

Write a raspberry PI python program to blink the LED with delay of 1 second.

Apparatus:

- Raspberry PI 4,
- Breadboard,
- LED,
- 220 ohm resistor,
- Jumper wires (Male to female),
- USB mouse, keyboard and Monitor if PI system is independent,
- VNC viewer if PI system is connected to network

Circuit Diagram:



```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
```

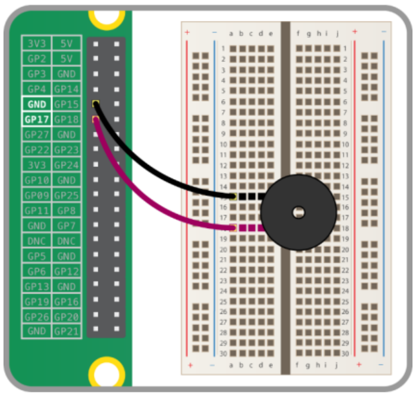
```
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set initial value to low (off)

while True: # Run forever
        GPIO.output(8, GPIO.HIGH) # Turn on
        sleep(1) # Sleep for 1 second
        GPIO.output(8, GPIO.LOW) # Turn off
        sleep(1) # Sleep for 1 second
```

Write a raspberry PI python program to sound the buzzer (ON and OFF) with delay of 1 second.

Apparatus:

- Raspberry PI 4,
- Breadboard,
- Buzzer,
- Jumper wires (Male to female),
- USB mouse, keyboard and Monitor if PI system is independent,
- VNC viewer if PI system is connected to network

Circuit Diagram:



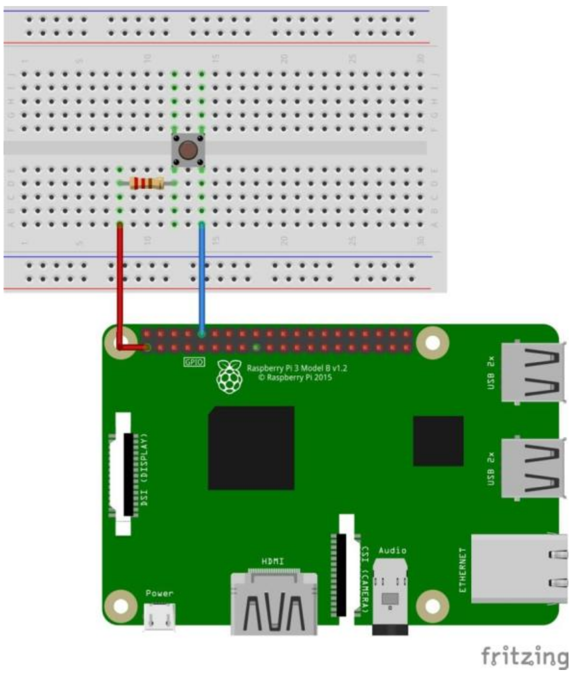
- Positive end of Buzzer is connected with PIN 11 on upper side (right side).
- Negative end of Buzzer is connected with PIN 9 (ground pin) on upper side (right side) which is ground pin.
- No need to installGPIO library, if it is already installed.
- In our program following steps must be executed
  - Initialize GPIO Ports

- Turn the Buzzer ON and OFF
- Provide delay of 1 second in between.

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(11, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set
initial value to low (off)

while True: # Run forever
        GPIO.output(11, GPIO.HIGH) # Turn on
        sleep(1) # Sleep for 1 second
        GPIO.output(11, GPIO.LOW) # Turn off
        sleep(1) # Sleep for 1 second
```

Interface Raspberry Pi with Push Button switch. If a button is presed then display a message. ie "Push button switch is ON".

Circuit Diagram:



```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Set pin 10 to
be an input pin and set initial value to be pulled low (off)

while True:
        if GPIO.input(10) == GPIO.high:
                print("Button is pushed")
```

IoT Applications based on Raspberry PI Processor

1. IoT based Smart Garden:
    - Measure and Control environmental parameters like temperature, humidity, wind capacity and soil moisture.
    - Through smart phone above parameters can be controlled.
    - For example, if soil is dry then start supplying water.
    - If temperature or rain is high, then you can cover the garden using green net.
2. An industrial IoT Controller:
    - Employee attendance through smart phone / wireless devices located in industry range.
    - To maintain and control fire safety devices which sends the alerts in case of fire to your smart phone.
    - Smart parking for employees.
3. Air Quality Monitor:
    - Your IoT devices can measure temperature and humidity from the air.
    - Various gases including oxygen, carbon or other smokes can be measured.
    - Based on the amount of gases alerts and messages can be sent to users.
4. IoT based Smart Energy Monitor
    - This system can measure the energy / power unit produced from solar system and the information will be sent to owner through mobile applications.
    - Even the power consumed during day / night time can be measured and controlled using smart phones.
5. Smart doorbell and video intercom system:
    - In this system camera located with doorbell can capture the image of a person and on other side the person who is opening the door can able to see who is standing at the door.
6. IoT based home automation using Raspberry PI
7. IoT based smart parking system for Malls / complex.
8. IoT based smart vacuum cleaner
9. Health Monitoring system using Raspberry PI
10. IoT based smart dustbins.

PYTHON PROGRAMMING

Why to choose python for Raspberry PI programming?

- Python is simple because it is having simple syntax similar to English language.
- Python is free. It does not require any license. (open source license)
- It is free to install, use, and distribute, even for commercial purposes.
- Python is platform independent. It can work on any OS like windows, Linux or MAC.
- Python is portable because code or program written on one platform can work on other platform too.
- Python is interpreted.
- Python is popular.

What is the output of following python code lines?

1. print(0b1011)
2. print(0o10)
3. print(0xFF)
4. print(0b1011110)
5. print(0o201)
6. print(0xFC)

It will always print decimal number.

print(0b1011) will print decimal value of binary number $(1011)_2$.
It will print 11.

8	4	2	1
1	0	1	1

Print(0o10) will print decimal value of octal number $(10)_8$.
(10) octal in decimal = $1x8^1 + 0x8^0 = 8 + 0 = 8$

Print(0xFF) will print decimal value of hexadecimal number $(FF)_{16}$.
(FF) hexadecimal in decimal = $Fx16^1 + Fx16^0 = 15x16 + 15x1 = 240 + 15 = 255$

print(0b1011110) will print decimal value of binary number $(1011110)_2$.
It will print (64 +16 + 8 + 4 + 2 = 94).

64	32	16	8	4	2	1

1	0	1	1	1	1	0

Print(0o201) will print decimal value of octal number $(201)_8$.
(201) octal in decimal = 2x 8^2 +0x8^1 + 1x8^0 = 2x 64 + 0 + 1 x1 = 128 + 1 = 129

Print(0xFC) will print decimal value of hexadecimal number $(FC)_{16}$.
(FC) hexadecimal in decimal = Fx16^1 + Cx16^0 = 15x16 + 12x1 = 240 + 12 = 252

Explain python operators used in Raspberry PI programming.

Operators are used for performing various operations on numbers and variables. Python offers following seven categories of operators.

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Arithmetic Operators
Arithmetic operators are used to perform simple mathematical operations on numeric values (except complex). (+ , -, / ,*, %, //, **)

Assignment Operators (=, +=, -=,/=,*=)
Assignment operators are used to assign new values to variables.

Comparison Operators (>, <, <= , >=, !=)
Comparison operators are used to compare two values.

Logical Operators (AND, OR, NOT)
Logical operators are used to join two or more conditions.

Identity Operators (is , isnot)
Identity operators are used to check if two objects point to the same object, with the same memory location.

What is List in python? How to create list in Python?

<u>Introduction to list:</u>

- A list is a sequence of values.
- It is similar to array.
- Values in the list are known as items / elements.

L=[1,2,3]

- List is an ordered thing.
- It remembers the order of items inserted.
- Items in the list are accessed by an index.
- List can have any type of data like numbers and string.
- List can be nested.
- List is changeable means user can add, remove or replace any items / values.
- A list can have zero items in it. Such type of list is known as an empty list.
- It can be create by simply typing square brackets without any values.

L=[]

- User can create list using List() constructor.
- List() constructor is used to convert other data type into list.
- To create a list “list comprehension” method can be used.

What do you mean by nested list? How to create it?

<u>Nested List:</u>

A list can contain sub lists, which in turn can contain sub lists themselves, and so on. This is known as nested list.

L = ['a', ['bb', ['ccc', 'ddd'], 'ee', 'ff'], 'g', 'h']

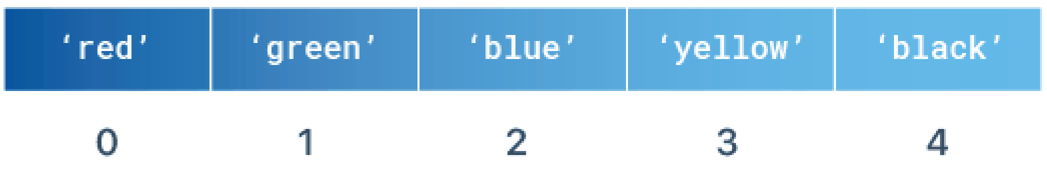
To create the list use above method, where a list items are written inside square brackets. Foe nested list use another square brackets to create sub list.

Explain Positive indexing and negative indexing methds to access items from the list. (IMP)

Positive Indexing and negative indexing are used to access items from the list. There is a mapping between values and index. Each index can have fix value from the list.

Positive Indexing:

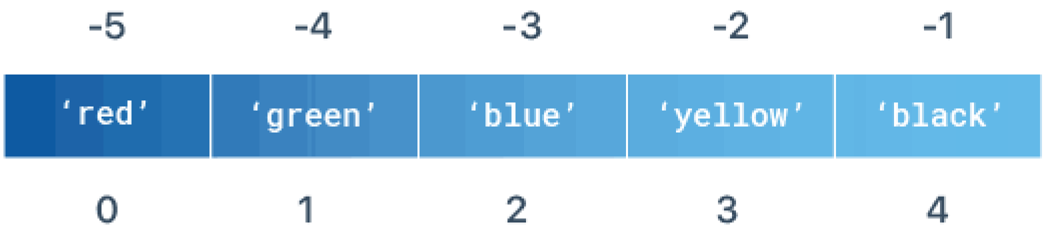
L = ['red', 'green', 'blue', 'yellow', 'black']



Index always start with '0'. So if you want to print all items in the list by index then use following code.

Print(L[0])→red
Print(L[1])→green
Print(L[2])→blue
Print(L[3])→yellow
Print(L[4])→black
Print(L[5])→it displays index error

Negative List Indexing:



Positive indexing starts from left to right and it includes '0'.
Negative indexing starts from right to left and do not include '0'.

Negative indexing counts from backward or end of the list.

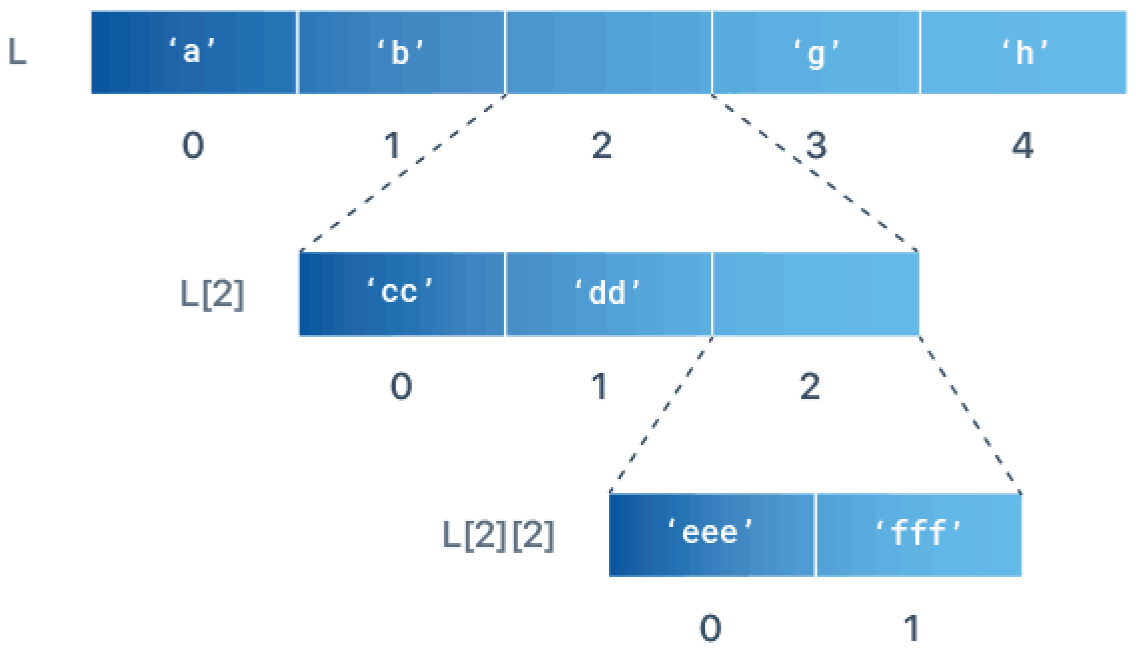
Print(L[-1])→Black
Print(L[-2])→yellow
Print(L[-3])→blue
Print(L[-4])→green
Print(L[-5])→red

L[0] = L[-5] = Red
L[1] = L[-4] = Green
L[2] = L[-3] = Blue
L[3] = L[-2] = Yellow

L[4] = L[-1] = Black

with example explain how to access items from the nested list?

Given List L = ['a', 'b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']



How to print 'a'→print(L[0])
How to print 'b'→print(L[1])
How to print 'g'→print(L[3])
How to print 'h'→print(L[4])

How to print 'cc'→print(L[2][0])
How to print 'dd'→print(L[2][1])

How to print 'eee'→print(L[2][2][0])
How to print 'fff'→print(L[2][2][1])

What do you mean by slicing? How to create slice in the list?

Slice:

- A segment of a list is known as slice.
- Slice is also a list.

- The slice operator [n:m] returns the part of the list from the "nth " item to the "mth" item, including the first(n) but excluding the last(m).

Creating slice:

L = ['a', 'b', 'c', 'd', 'e', 'f']

-6	-5	-4	-3	-2	-1
a	b	c	d	e	f
0	1	2	3	4	5

Print L[0:6]→[a,b,c,d,e,f]
Print L[-5:-2]→[b,c,d]
print(L[2:5])→ ['c', 'd', 'e']
print(L[0:2])→ ['a', 'b']
print(L[3:-1])→Prints ['d', 'e']

What do you mean by python Tuple? How to create it?

Tuple definition:

- A tuple is an ordered collection of values.

- A tuple is similar to List in many ways.
- Items in a tuple can be accessed by an index.
- Tuple contains any sort of objects like numbers, character, Boolean, list and even tuple.
- **Tuples are immutable means it cannot be editable**. User can not add, remove or modify the values.

How to create Tuple?

- It is created by placing comma separated value in parentheses ().

```
# A tuple of integers
T = (1, 2, 3)

# A tuple of strings
T = ('red', 'green', 'blue')

# A tuple with mixed datatypes
T = (1, 'abc', 1.23, True)
```

```
# An empty tuple
T = ()
```

A tuple containing zero items is known as empty tuple.

```
# A tuple without parentheses
T = 1, 'abc', 1.23, True
```

Tuple doesn’t require parentheses. It is just a list separated by comma.

Explain how tuple is created using tuple() constructor.
Or
How to convert any list or string into tuple?

The Tuple() Constructor:

User can convert other data type into tuple using tuple() constructor.

```
# Convert a list to a tuple
T = tuple([1, 2, 3])
print(T)
# Prints (1, 2, 3)

# Convert a string to a tuple
T = tuple('abc')
print(T)
# Prints ('a', 'b', 'c')
```

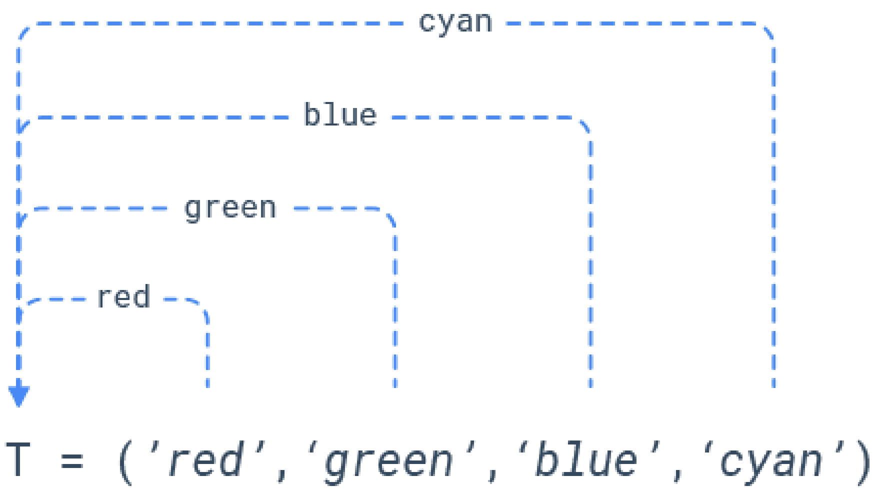
Explain Tuple Packing and Tuple unpacking. Explain the use of packing and unpacking of tuple.

Tuple packing:

When a tuple is created, the items in the tuple are packed together into the object.

```
T = ('red', 'green', 'blue', 'cyan')
```

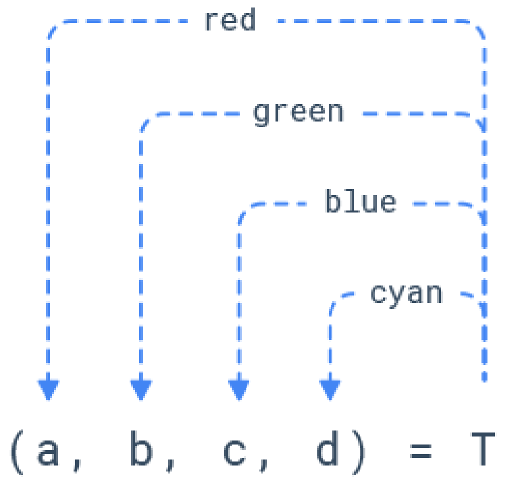
In above example, the values ‘red’, ‘green’, ‘blue’ and ‘cyan’ are packed together in a tuple.



Tuple Unpacking:
When a packed tuple is assigned to a new tuple, the individual items are unpacked (assigned to the items of a new tuple).

```
 T = ('red', 'green', 'blue', 'cyan')
(a, b, c, d) = T
```

In above example T is unpacked to 4 variables a, b, c, d.



Use of packing / unpacking concept:
When you want to swap two variables without use of temporary variable then this concept is used.

Give the comparison between List and Tuple. (IMP)

	List	**Tuple**
1	List items can be changed.	Tuple items cannot be changed.
2	List is created by []. (square bracket)	Tuple is created by (). (round bracket)
3	List items can be accessed by index.	Tuple items can be accessed by index.
4.	Nested list is possible.	Nested Tuple is also possible.
5	L=[1,2,3,4]	T=(1,2,3,4)
6.	List is mutable	Tuple is immutable
7.	Packing and unpacking concept is not used in the list.	Packing and unpacking concept is used in the Tuple.

What is set? Explain Python Set and its properties. (5 marks)

Set
Set is collection of meaningful data.

Set is unordered means we can arrange data in any order.
A={1,2,3,4,5}
A={5,4,3,2,1}
A={3,2,1,5,4}

To create the set we are using curly brackets { }.

A={1,1,1,2,3}
It is not a set because it contains duplicate items.

Set doesn't have any duplicate data.

Python set is unordered collection of unique items.

Whenever you want to perform any mathematical operations such as union , intersection, difference, symmetric difference etc. then you must require to use python set.

Properties of Python Sets:

1. Python sets are unordered. (It is not necessary to store items in particular order).
2. Python Sets items are unique. (Duplicate items are not allowed.)
3. Python sets are unindexed. (Items cannot be accessed by particular index.)
4. Python sets are changeable (mutable). (items from the sets can be removed, added or changed)

Explain various set operations used with python set.

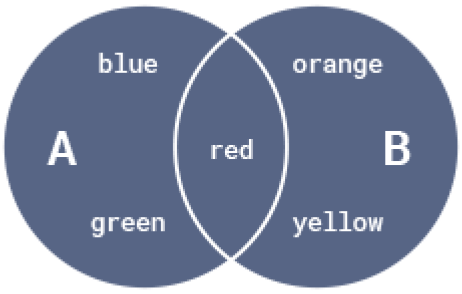
Set Operations:

1. Union operation
2. Intersection operation
3. difference operation
4. Symmetric difference operation

1. Union Operation:

A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}

A U B = {red, green, blue, yellow, orange}

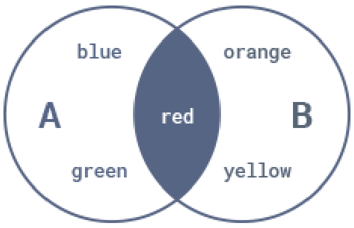


```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}

# by operator
print(A | B)
# Prints {'blue', 'green', 'yellow', 'orange', 'red'}

# by method
print(A.union(B))
# Prints {'blue', 'green', 'yellow', 'orange', 'red'}
```

2. Intersection Operation:

A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}

A (intersection) B = {red}



```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}

# by operator
print(A & B)
# Prints {'red'}

# by method
print(A.intersection(B))
# Prints {'red'}
```

3. Difference Operation:

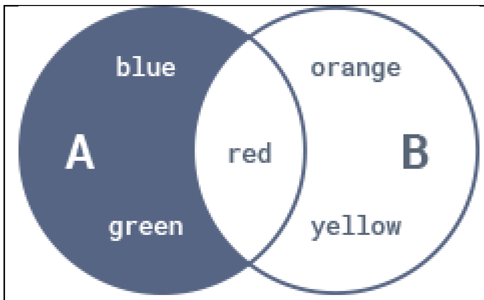
Set difference operation can be performed by subtracting members of one set from others.

A – B → It means remove members of B from A

A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}

A – B = {green, blue} = A – (A intersection B)
B – A = {yellow, orange} = B – (A intersection B)

It can be performed by "-" operator or difference method.



```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}
```

```
# by operator
print(A - B)
# Prints {'blue', 'green'}
```

```
# by method
print(A.difference(B))
# Prints {'blue', 'green'}
```

```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}
```

```
# by operator
print(B - A)
# Prints {'yellow', 'orange'}
```

```
# by method
print(B.difference(A))
# Prints {'yellow', 'orange'}
```

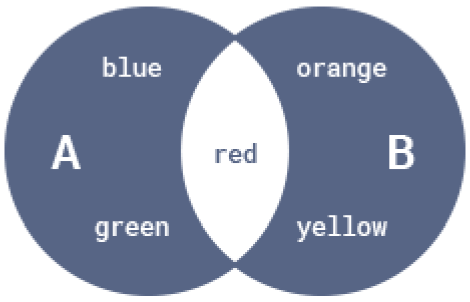
4. Symmetric Difference:

Set A and Set B

The set of all elements which belongs to either A or B but doesn’t belongs to both.

It can be performed by ‘^’ operator or symmetric_difference() method.

A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}



Dark portion in above figure represents symmetric difference output.

A^B = {blue, green, orange, yellow}

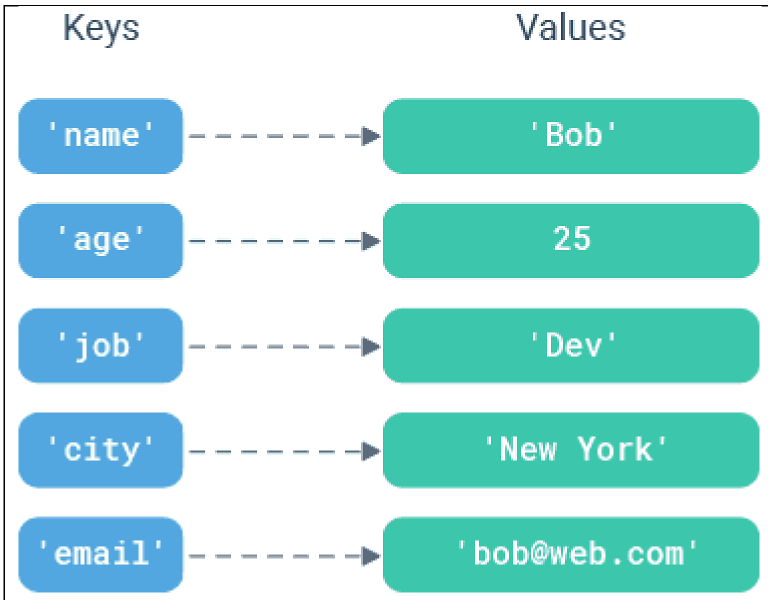
```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}

# by operator
print(A ^ B)
# Prints {'orange', 'blue', 'green', 'yellow'}

# by method
print(A.symmetric_difference(B))
# Prints {'orange', 'blue', 'green', 'yellow'}
```

What is Python Dictionary? How to create it?

Dictionaries are python implementation of data structure. It is also known as associative array.

Dictionary can be considered as set of indexes and set values.

Each key is mapped with specific values.
Association of key and its value is known as (key:value) pair.



<u>How to create dictionary?</u>

```
# Create a dictionary to store employee record
D = {'name': 'Bob',
    'age': 25,
    'job': 'Dev',
    'city': 'New York',
    'email': 'bob@web.com'}
```

Step-1: start writing in set { }
Step-2: write down the key within single quote(‘ ‘)
Step-3: separate key and its value by colon (:)
Step-4: type the value after column within single quote(‘ ‘)
Step-5: Insert the new records separated by coma. (,)

Using dict() constructor we can create the dictionary.

<u>Dictionary with List of two items Tuple:</u>

```
L = [('name', 'Bob'),
    ('age', 25),
    ('job', 'Dev')]

D = dict(L)
print(D)
# Prints {'name': 'Bob', 'age': 25, 'job': 'Dev'}
```

Dictionary with Tuple of two items List:

```
T = (['name', 'Bob'],
    ['age', 25],
    ['job', 'Dev'])

D = dict(T)
print(D)
# Prints {'name': 'Bob', 'age': 25, 'job': 'Dev'}
```

Explain any Four methods to create dictionary.

There are Four methods to create dictionary.

1. Direct method
2. Dictionary with list of two items tuple
3. Dictionary with tuple of two items list
4. Use zip() function with dict()

1. Direct Method

```
# Create a dictionary to store employee record
D = {'name': 'Bob',
    'age': 25,
    'job': 'Dev',
    'city': 'New York',
    'email': 'bob@web.com'}
```

Step-1: start writing in set { }
Step-2: write down the key within single quote(‘ ‘)
Step-3: separate key and its value by colon (:)
Step-4: type the value after column within single quote(‘ ‘)
Step-5: Insert the new records separated by coma. (,)

Using dict() constructor we can create the dictionary.

2. Dictionary with List of two items Tuple:

```
L = [('name', 'Bob'),
    ('age', 25),
    ('job', 'Dev')]

D = dict(L)
print(D)
```

```
# Prints {'name': 'Bob', 'age': 25, 'job': 'Dev'}
```

3. Dictionary with Tuple of two items List:

```
T = (['name', 'Bob'],
    ['age', 25],
    ['job', 'Dev'])

D = dict(T)
print(D)
# Prints {'name': 'Bob', 'age': 25, 'job': 'Dev'}
```

(For details refer que-29)

4. Use zip() and dict():

```
keys = ['name', 'age', 'job']
values = ['Bob', 25, 'Dev']

D = dict(zip(keys, values))

print(D)
# Prints {'name': 'Bob', 'age': 25, 'job': 'Dev'}
```

List and explain important properties of dictionary.

Important Properties of a Dictionary

Dictionaries are pretty straightforward, but here are a few points you should be aware of when using them.

1. Keys must be unique:

A key can appear in a dictionary only once.

Even if you specify a key more than once during the creation of a dictionary, the last value for that key becomes the associated value.

```
D = {'name': 'Bob',
    'age': 25,
    'name': 'Jane'}
print(D)
# Prints {'name': 'Jane', 'age': 25}
```

Notice that the first occurrence of ‘name’ is replaced by the second one.

2. Key must be immutable type:

You can use any object of immutable type as dictionary keys – such as numbers, strings, booleans or tuples.

```
D = {(2,2): 25,
    True: 'a',
    'name': 'Bob'}
```

An exception is raised when mutable object is used as a key.

```
# TypeError: unhashable type: 'list'
D = {[2,2]: 25,
    'name': 'Bob'}
```

3. Value can be of any type:

There are no restrictions on dictionary values. A dictionary value can be any type of object and can appear in a dictionary multiple times.

```
# values of different datatypes
D = {'a':[1,2,3],
    'b':{1,2,3}}

# duplicate values
D = {'a':[1,2],
    'b':[1,2],
    'c':[1,2]}
```