1. **Write the difference between dictionary and tuples?**

Ans:

Feature	List	Tuple	Dictionary
Definition	Ordered collection	Ordered, immutable collection	Unordered key-value pairs
Syntax	[]	()	{key: value}
Mutability	Mutable	Immutable	Mutable
Duplicates	Allowed	Allowed	Keys must be unique; values can repeat
Access	Access by index	Access by index	Access by key
Use Case	Dynamic collections	Fixed collections	Storing related data (key-value pairs)
Example	my_list = [1, 2, 3, "apple", 3.14]	my_tuple = (1, 2, 3, "apple", 3.14)	my_dict = {"name": "Alice", "age": 30, "city": "New York"}

2. **Explain the different dictionary methods with examples.**

Ans:

keys(), values(), and items() are three dictionary methods that will return list-like values of the dictionary's keys, values, or both keys and values: The values returned by these methods are not true lists: they cannot be modified and do not have an append() method.

```
spam = {'color': 'red', 'age': 42}
for v in spam.values():
    print(v)
```
output:

```
red
42
```
```
spam = {'color': 'red', 'age': 42}
for k in spam.keys():
    print(k)
```
output:

```
color
age
```
```
spam = {'color': 'red', 'age': 42}
for i in spam.items():
    print(i)
```
output:

```
('color', 'red')
('age', 42)
```
Checking Whether a Key or Value Exists in a Dictionary using in and not in operator

```
>>> spam = {'name': 'Zophie', 'age': 7}
>>> 'name' in spam.keys()
 True
 >>> 'Zophie' in spam.values()
 True
 >>> 'color' in spam.keys()
 False
```
The get() Method

The get() method returns the value of the item with the specified key.

Syntax

dictionary.get(*keyname, value*)

```
>>>d={'name':'arjun','age':20,'usn':'1si24me001'}
>>>d.get('usn')
'1si24me001'
```

```
>>>d.get('branch','mech') # key is not there in dictionary, specificy the value as second argument to be returned
'mech'
```

The setdefault() Method

The setdefault() method returns the value of the item with the specified key. If the key does not exist, insert the key, with the specified value

Syntax

dictionary.setdefault(*keyname, value*)

```
>>>d={'name':'arjun','age':20,'usn':'1si24me001'}
>>>d.setdefault('name')
'arjun'
>>>d.setdefault('branch','mech')
'mech'
>>>d
{'name': 'arjun', 'age': 20, 'usn': '1si24me001', 'branch': 'mech'}
```

3. **Write the python program to add birthday of friend using birthday as dictionary with names as keys and birthday as values.**

Ans:

```
#dictionary with the names as keys and the birthdays as values
birthdays = {'Alice': 'Apr 1', 'Bob': 'Dec 12', 'Carol': 'Mar 4'}
while True:        #infinite while loop to check the name in birthday dictionary.
   print('Enter a name: (blank to quit)')        #message to print for user
   name = input()                                # taking input from user using input()
   if name == '':
      break
   if name in birthdays:                         # if name in birthdays dictionary
      print(birthdays[name] + ' is the birthday of ' + name)
   else:                                         # if name not in birthdays dictionary
      print('I do not have birthday information for ' + name)
      print('What is their birthday?')
#Add it using the same square bracket syntax combined with the assignment operator
      birthdays[name] = input()
      print('Birthday database updated.')
```

Output:

```
Enter a name: (blank to quit)
Alice
Apr 1 is the birthday of Alice
Enter a name: (blank to quit)
abhay
I do not have birthday information for abhay
What is their birthday?
Jan 1
Birthday database updated.
Enter a name: (blank to quit)
abhay
Jan 1 is the birthday of abhay
Enter a name: (blank to quit)
```

4. Explain the different useful string methods.

Ans:

The upper(), lower(), isupper(), and islower() Methods

The upper() and lower() string methods return a new string where all the letters in the original string have been converted to uppercase or lowercase, respectively.

```
>>> spam = 'Hello, world!'
 >>> spam = spam.upper()
>>> spam
 'HELLO, WORLD!'
>>> 'HELLO'.isupper()
 True
 >>> 'abc12345'.islower()
 True
```

The isX() Methods

isalpha() Returns True if the string consists only of letters and isn’t blank
isalnum() Returns True if the string consists only of letters and numbers is not blank

isdecimal() Returns True if the string consists only of numeric character and is not blank

isspace() Returns True if the string consists only of spaces, tabs, and newlines and is not blank

istitle() Returns True if the string consists only of words that begin with an uppercase letter followed by only lowercase letters

```
>>> 'hello'.isalpha()
 True
 >>> 'hello123'.isalpha()
 False
 >>> 'hello123'.isalnum()
 True
 >>> 'hello'.isalnum()
 True
 >>> '123'.isdecimal()
 True
 >>> '    '.isspace()
 True
 >>> 'This Is Title Case'.istitle()
 True
```

The startswith() and endswith() Methods

The startswith() and endswith() methods return True if the string value they are called on begins or ends (respectively) with the string passed to the method; otherwise, they return False.

```
>>> 'Hello, world!'.startswith('Hello')
 True
 >>> 'Hello, world!'.endswith('world!')
 True
```

The join() and split() Methods

The join() method is useful when you have a list of strings that need to be joined together into a single string value. The join() method is called on a string, gets passed a list of strings, and returns a string.

```
>>> ', '.join(['cats', 'rats', 'bats'])
 'cats, rats, bats'
>>> 'My name is Simon'.split()
 ['My', 'name', 'is', 'Simon']
```

Splitting Strings with the partition() Method

The partition() string method can split a string into the text before and after a separator string.

```
>>> 'Hello, world!'.partition('w')
('Hello, ', 'w', 'orld!')
```

Justifying Text with the rjust(), ljust(), and center() Methods

The rjust() and ljust() string methods return a padded version of the string they are called on, with spaces inserted to justify the text. The first argument to both methods is an integer length for the justified string.

```
>>> 'Hello'.rjust(10)
'     Hello'
>>> 'Hello'.ljust(10)
'Hello     '
>>> 'Hello'.rjust(20, '*')
'***************Hello'
>>> 'Hello'.center(20)
'       Hello        '
```

Removing Whitespace with the strip(), rstrip(), and lstrip() Methods

The strip() string method will return a new string without any whitespace characters at the beginning or end. The lstrip() and rstrip() methods will remove whitespace characters from the left and right ends, respectively.

```
>>> spam = '    Hello, World    '
>>> spam.strip()
'Hello, World'
>>> spam.lstrip()
'Hello, World    '
>>> spam.rstrip()
'    Hello, World'
```

5. **Write a python program to count number of characters in a given string using setdeafult() method.**

Ans:

The program loops over each character in the variable a assigned string message, counting how often each character(i) appears. The setdefault() method call ensures that the key is in the count dictionary (with a default value of 0).

```
a = 'All the best'
```

```
count = {}
for i in a:
   count.setdefault(i, 0)
   count[i] = count[i] + 1
print(count)
```

output:

```
{'A': 1, 'l': 2, ' ': 2, 't': 2, 'h': 1, 'e': 2, 'b': 1, 's': 1}
```

6. **Write a python program to create folders, folder and delete the folder using os module.**

Ans:

To create new folders (directories) with the os.makedirs() function.

```
>>> import os
>>> os.makedirs('C:\\Users\\pc\\Desktop \\delicious\\walnut\\waffles')
```

To create new folder use os.mkdir(path)

```
>>> import os
>>>os.mkdir('C:\\Users\\pc\\Desktop\mech')
```

To remove folder use os.rmdir(path0

```
>>> import os
>>>os.rmdir('C:\\Users\\pc\\Desktop\mech')
```

To delete file use os.remove(path)

```
>>> import os
>>>os.remove('C:\\Users\\pc\\Desktop\spam.txt')
```

7. **Write a python program to find the size of file, folder using os module.**

Ans:

The os.path module provides functions for finding the size of a file in bytes and the files and folders inside a given folder.

- Calling os.path.getsize(path) will return the size in bytes of the file in the path argument.
- Calling os.listdir(path) will return a list of filename strings for each file in the path argument.

```
>>>import os
>>>os.path.getsize("C:\\Users\\pc\\Desktop\\spam.txt")
3677
>>>os.listdir('C:\\Users\\pc\\Desktop\\spam')
['Fluid Mechanis and Machinery S3MEI01.docx', 'FM Evening Test2 Question Paper.pdf', 'hello.txt', 'New Microsoft Word Document.docx', 'plc2.txt', 'Question bank Fluid mechanics amf Machinery S3MEI01.docx', 'Question bank Quiz2 PLC2 1st semester 2024-25.docx', 'spam.txt', 'thermo-rk-rajput.pdf']
```

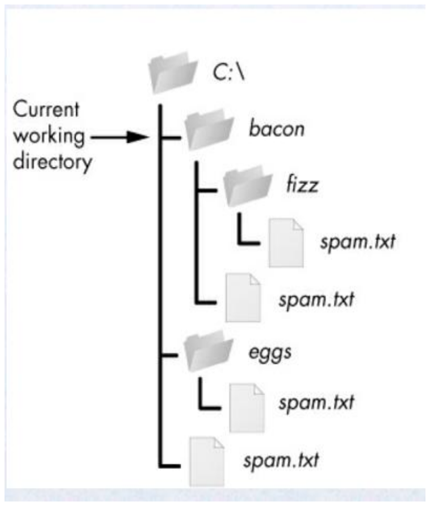
call os.path.getsize() and os.path.join() to join the folder name with the current filename. The integer that os.path.getsize() returns is added to the value of totalSize. After looping through all the files, totalSize is size of the folder

```
#to find size of folder
import os
totalSize = 0
for filename in os.listdir('C:\\Users\\pc\\Desktop\\spam'):
   totalSize = totalSize
+os.path.getsize(os.path.join('C:\\Users\\pc\\Desktop\\spam', filename))
print(totalSize)
```

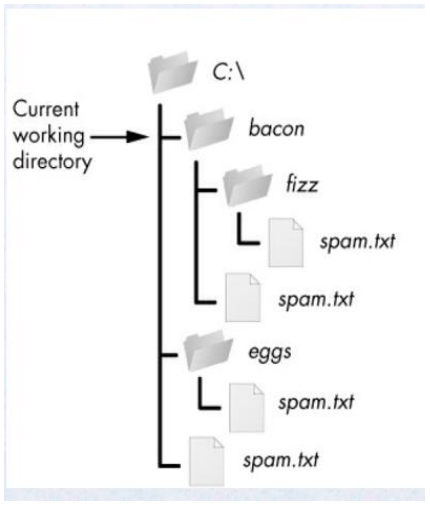
output:

```
6429188
```

8. **What is absolute and relative path. Write the absolute and relative path for the directory.**



- There are two ways to specify a file path:
  - ✓ An **absolute path**, which always begins with the root folder
  - ✓ A **relative path**, which is relative to program's cwd.
- A single period **.** (“**dot**”) =“this directory.” (**CWD**)
- Two periods **..** (“**dot dot**”) =“**the parent folder**(directory).”
- When cwd is set to C:\bacon, relative paths for other folders & files are set as : (The .\ at start of a relative path is optional)

	**Absolute Path**	**Relative Path**
	C:\	.. Or ..\
	C:\ bacon	. Or .\
	C:\bacon\fizz	. \ fizz
	C:\ bacon\ fizz\spam.txt	. \ fizz\spam.txt
	C:\ bacon\spam.txt	. \ spam.txt
	C:\ eggs	.. \ eggs
	C:\ eggs \ spam.txt	.. \ eggs\spam.txt
	C:\ spam.txt	.. \spam.txt

9. Write a python program to open the file, read the contents of the file and write the contents in to the files.

Ans:

The more common way of writing to a file involves using the open() function and file objects. There are three steps to reading or writing files in Python:

1. Call the open() function to return a File object.

2. Call the read() or write() method on the File object.

3. Close the file by calling the close() method on the File object.

Opening Files with the open() Function

To open a file with the open() function, you pass it a string path indicating the file you want to open; it can be either an absolute or relative path. The open() function returns a File object.

```
# create a spam.txt file with contents
Hello
How are you?
I am fine
>>>f1=open('C:\\Users\\pc\\Desktop\\spam.txt')
>>>f1.read()                                   # read all contents of file
'Hello\nHow are you?\nI am fine\n'
>>>f1.close()
>>>f1=open('C:\\Users\\pc\\Desktop\\spam.txt','r')
>>>f1.read(5)                                  # read first 5 characters
'Hello'
>>>f1.close()
>>>f1=open('C:\\Users\\pc\\Desktop\\spam.txt','r')
>>>f1.readline()                               # read first line
'Hello\n'
>>>f1.close()
>>>f1=open('C:\\Users\\pc\\Desktop\\spam.txt','r')
>>>f1.readlines()                              # reads all lines
['Hello\n', 'How are you?\n', 'I am fine\n']
>>>f1.close()
```

To write the contents in to the file in "write plaintext" mode or "append plaintext" mode, or write mode and append mode.

```
>>>f1=open('C:\\Users\\pc\\Desktop\\hello.txt','w') # It opens the file
hello.txt at the path and if file already exit with contents in it it will get
erased.
>>>f1.write('Hello world!')     # it returns number of characters written on hello.txt file
12
>>>f1.close()
>>>f1=open('C:\\Users\\pc\\Desktop\\hello.txt','r')
>>>f1.read()
'Hello world!'
>>>f1=open('C:\\Users\\pc\\Desktop\\hello.txt','a')
>>>f1.write('sit')                         # write the contents to end in append mode
3
>>>f1.close()
>>>f1=open('C:\\Users\\pc\\Desktop\\hello.txt','r')
>>>f1.read()
'Hello world!sit'
```

10. Explain the different shutil module operations.

Ans:

The shutil module provides functions for copying files, as well as entire folders.

Calling shutil.copy(source, destination) will copy the file at the path source to the folder at the path destination. (Both source and destination can be strings or Path objects.). This function returns a string or Path object of the copied file.

```
>>>import shutil
>>>shutil.copy('C:\\Users\\pc\\Desktop\\spam.txt','C:\\Users\\pc\\Deskt
op\\spam')
'C:\\Users\\pc\\Desktop\\spam\\spam.txt'
```

While shutil.copy() will copy a single file**, shutil.copytree()** will copy an entire folder and every folder and file contained in it. Calling shutil.copytree(source, destination) will copy the folder at the path source, along with all of its files and subfolders, to the folder at the path destination. The source and destination parameters are both strings. The function returns a string of the path of the copied folder.

```
>>>import shutil
```

```
>>>shutil.copytree('C:\\Users\\pc\\Desktop\\spam','C:\\Users\\pc\\Desktop\\spam1')
'C:\\Users\\pc\\Desktop\\spam1'
```

Calling shutil.move(source, destination) will move the file or folder at the path source to the path destination and will return a string of the absolute path of the new location.

```
>>>import shutil
>>>shutil.move('C:\\Users\\pc\\Desktop\\hello.txt','C:\\Users\\pc\\Desktop\\spam1')
'C:\\Users\\pc\\Desktop\\spam1\\hello.txt'
>>>import shutil,os
>>>shutil.move('C:\\Users\\pc\\Desktop\\hello.txt','C:\\Users\\pc\\Desktop\\spam1\\hello1.txt')
'C:\\Users\\pc\\Desktop\\spam1\\hello1.txt'
```

Permanently Deleting Files and Folders

To delete a single file or a single empty folder with functions in the os module, whereas to delete a folder and all of its contents, you use the shutil module.

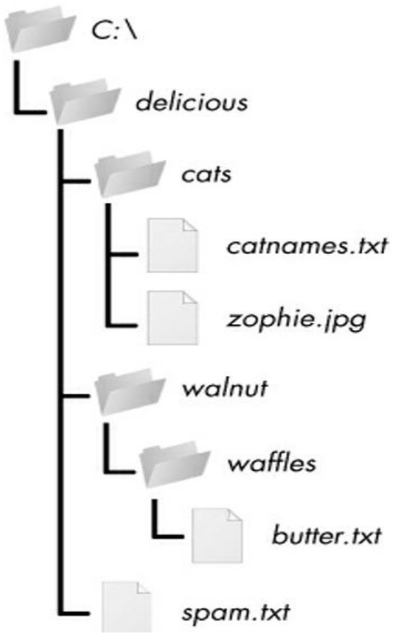
Calling os.unlink(path) will delete the file at path.

Calling os.rmdir(path) will delete the folder at path. This folder must be empty of any files or folders.

Calling shutil.rmtree(path) will remove the folder at path, and all files and folders it contains will also be deleted.

```
>>>import shutil,os
>>>os.unlink("C:\\Users\\pc\Desktop\\spam.txt")
>>>os.rmdir("C:\\Users\\pc\Desktop\\spam")
>>>shutil.rmtree("C:\\Users\\pc\Desktop\\spam1")
```

11. Write a python program to display the path of folder, subfolders and files in a directory using os.walk().



The os.walk() function is passed a single string value: the path of a folder. The use os.walk() in a for loop statement to walk a directory tree, much like how you can use the range() function to walk over a range of numbers.
The os.walk() function will return three values on each iteration through the loop:
A string of the current folder's name
A list of strings of the folders in the current folder
A list of strings of the files in the current folder

```
import os
for folderName, subfolders, filenames in
os.walk('C:/Users/pc/Desktop/delicious'):
  print('The current folder is ' + folderName)
  for subfolder in subfolders:
    print('SUBFOLDER OF ' + folderName + ': ' + subfolder)
  for filename in filenames:
    print('FILE INSIDE ' + folderName + ': '+ filename)
  print('')
```

output:

```
The current folder is C:/Users/pc/Desktop/delicious
SUBFOLDER OF C:/Users/pc/Desktop/delicious: cats
SUBFOLDER OF C:/Users/pc/Desktop/delicious: walnut
FILE INSIDE C:/Users/pc/Desktop/delicious: spam.txt
```

```
The current folder is C:/Users/pc/Desktop/delicious\cats
FILE INSIDE C:/Users/pc/Desktop/delicious\cats: catnames.txt
FILE INSIDE C:/Users/pc/Desktop/delicious\cats: zophie.bmp

The current folder is C:/Users/pc/Desktop/delicious\walnut
SUBFOLDER OF C:/Users/pc/Desktop/delicious\walnut: waffles

The current folder is C:/Users/pc/Desktop/delicious\walnut\waffles
FILE INSIDE C:/Users/pc/Desktop/delicious\walnut\waffles: butter.txt
```