UNIT 3

Q.explain slicing, negative indexing and indexing of tuple.

Q.short notes on built in tuple.

1 cmp(tuple1,tuple2)-It compares two tuples and returns true if tuple1 is greater than tuple2

otherwise false.

2 len(tuple)-It calculates the length of the tuple.

3 max(tuple)-It returns the maximum element of the tuple

4 min(tuple-It returns the minimum element of the tuple.

5 tuple(seq)-It converts the specified sequence to the tuple.

Q.explain basic tuple operations.

Repetition -The repetition operator enables the tuple elements to be

repeated multiple times.

T1*2 = (1, 2, 3, 4, 5, 1, 2,

3, 4, 5)

Concatenation -It concatenates the tuple mentioned on either side of the operator.

T1+T2 = (1, 2, 3, 4, 5, 6,

7, 8, 9)

Membership -It returns true if a particular item exists in the tuple

otherwise false

print (2 in T1) prints True.

Iteration -The for loop is used to iterate over the tuple elements.

```
for i in T1:
 print(i)
```

Output

```
1
2
3
4
5
```

Length -It is used to get the length of the tuple. len(T1) = 5

Q.explain tuple and dictionary with program.

Q. explain bult in dictionary function.

1 cmp(dict1,dict2)-It compares the items of both the dictionary and returns true if the first dictionary values are greater than the second dictionary, otherwise it returnfalse.

2 len(dict)-It is used to calculate the length of the dictionary.

3 str(dict) -It converts the dictionary into the printable string representation.

4 type(variable) -It is used to print the type of the passed variable.

Q.explain any 5 built-in dictionary methods

1 dict.clear() It is used to delete all the items of the dictionary.

2 dict.copy() It returns a shallow copy of the dictionary.

3 dict.items() It returns all the key-value pairs as a tuple.

4 dict.keys() It returns all the keys of the dictionary.

5 dict.values() It returns all the values of the dictionary.

Q.explain any 5 built-in exception.

1 ArithmeticError- Raised when an error occurs in numeric calculations

2 FloatingPointError-Raised when a floating point calculation fails

3 ImportError-Raised when an imported module does not exist

4 KeyError-Raised when a key does not exist in a dictionary

5 MemoryError-Raised when a program runs out of memory

6 NameError-Raised when a variable does not exist

7 SyntaxError-Raised when a syntax error occurs

8 SystemExit-Raised when the sys.exit() function is called

Q.explain exception with arugments with eg.

An exception can have an argument, which is a value that gives additional information about the problem. The contents of the argument vary by exception. You capture an exception's argument by supplying a variable in the except clause as follows −

```
try:
   You do your operations here;
   ......................
except ExceptionType, Argument:
   You can print value of Argument here...
```

If you write the code to handle a single exception, you can have a variable follow the name of the exception in the except statement. If you are trapping multiple exceptions, you can have a variable follow the tuple of the exception.

This variable receives the value of the exception mostly containing the cause of the exception. The variable can receive a single value or multiple values in the form of a tuple. This tuple usually contains the error string, the error number, and an error location.

Example

Following is an example for a single exception −

```
#!/usr/bin/python
# Define a function here.
def temp_convert(var):
  try:
    return int(var)
  except ValueError, Argument:
    print "The argument does not contain numbers\n", Argument
# Call above function here.
temp_convert("xyz");
```

Q.what is file and what are its operating modes

Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).

Since Random Access Memory (RAM) is volatile (which loses its data when the computer is turned off), we use files for future use of the data by permanently storing them.

When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order:

Open a file

Read or write (perform operation)

Close the file

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.

x	Opens a file for exclusive creation. If the file already exists, the operation fails.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Opens in text mode. (default)
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

Q.short notes on exception handling.

Python has many built-in exceptions that are raised when your program encounters an error (something in the program goes wrong).

When these exceptions occur, the Python interpreter stops the current process and passes it to the calling process until it is handled. If not handled, the program will crash.

Exceptions: Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Try and Except Statement – Catching Exceptions

Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

Catching Specific Exception

A try statement can have more than one except clause, to specify handlers for different exceptions. Please note that at most one handler will be executed. For example, we can add IndexError in the above code. The general syntax for adding specific exceptions are –

```
try:
    # statement(s)
except IndexError:
    # statement(s)
except ValueError:
    # statement(s)
```

Try with Else Clause

In python, you can also use the else clause on the try-except block which must be present after all the except clauses. The code enters the else block only if the try clause does not raise an exception.

Finally Keyword in Python

Python provides a keyword finally, which is always executed after the try and except blocks. The final block always executes after normal termination of try block or after try block terminates due to some exception.

Syntax:

```
try:

   # Some Code....


except:

   # optional block

   # Handling of exception (if required)


else:

   # execute if no exception


finally:

   # Some code .....(always executed)
```

Raising Exception

The raise statement allows the programmer to force a specific exception to occur. The sole argument in raise indicates the exception to be raised. This must be either an exception instance or an exception class (a class that derives from Exception).

Q.what is directory. whic method are available to deal with directories in python.

All files are contained within various directories, and Python has no problem handling these too. The os module has several methods that help you create, remove, and change directories.

The mkdir() Method

You can use the mkdir() method of the os module to create directories in the current directory. You need to supply an argument to this method which contains the name of the directory to be created.

Syntax

```
os.mkdir("newdir")
```

eg-

```
#!/usr/bin/python

import os

# Create a directory "test"

os.mkdir("test")
```

The chdir() Method

You can use the chdir() method to change the current directory. The chdir() method takes an argument, which is the name of the directory that you want to make the current directory.

Syntax

```
os.chdir("newdir")
```

eg-

```
#!/usr/bin/python

import os

# Changing a directory to "/home/newdir"

os.chdir("/home/newdir")
```

The getcwd() Method

The getcwd() method displays the current working directory.

Syntax

```
os.getcwd()
```

eg-

```
#!/usr/bin/python
import os
# This would give location of the current directory
os.getcwd()
```

The rmdir() Method

The rmdir() method deletes the directory, which is passed as an argument in the method.

Before removing a directory, all the contents in it should be removed.

Syntax

```
os.rmdir('dirname')
```

eg-

```
#!/usr/bin/python

import os

# This would remove "/tmp/test" directory.

os.rmdir( "/tmp/test" )
```