**Module 5**

**Tips to Become a Better Programmer**

**1. Know Basics**

**You should have a good understanding of basics of the programming language in which you are working or learning. If your basic concepts are strong then you can solve complex problems by breaking it into smaller problems. Your code should be neat and well structured, so that if you start working with the code after few months then you can easily know where you left off.**

**2. Practice a Lot**

**We all know the fact that “practice makes a man perfect”. For improving your skills you have to practice thousands of hours. There are plenty of ways for practicing, like you can solve problems from good programming books or find online programming contests. There are many awesome websites like codechef.com, topcoder.com, etc where you can practice.**

**3. Be Amenable to Change**

**Technology is changing very rapidly so you have to aware of such changes. You should make sure that the projects or technologies on which you are working are not outdated. For becoming a better programmer you should always engage yourself in learning new things.**

**4. Read a Lot**

**The more you read, the more you learn. Read good programming books, articles, documentations. The best way for reading and learning programming is to join various programming forums and contests.**

**5. Help Others**

**Join great forums like stackoverflow.com where you can help other programmers by answering there questions. By helping others you can find answers of your own questions. You can also ask questions in the forums and get your problems solved; this will help you to become better programmer.**

**File Handling in python**

**Opening a file**

f = open(filename, mode)

Mode :

“r” : Read only

“w” : Writing - If file already exists it’s data will be cleared before opening.

“a” : Append

“wb” : Write in binary mode

“rb” : Read in binary mode

**Closing a file**

f.close()

Where f is file handler

**Reading Data**

For reading data from a file we can use the following methods

1. read([member])

Returns specified number of characters from a file. To read the entire contents of a file omit number.

f.read()

reads entire file

f.read(100)

reads only 100 characters

2. readline()

Return the next line of the file.

f.readline()

read next one line

3. readlines()

Return all next lines of a file

f.readlines()

**Appending Data**

f = open(‘sample.txt’,’a’)

f.write(“Hello friends\n”)

f.close()

Here we open the file in append mode and append a line ‘Hello friends\n’ using write method and close the file. In this case if the file ‘sample’ file already exists the new content will be appended to the existing content. If we open the same file with ‘w’ mode instead of ‘a’ mode the entire content will be deleted and appended only the new content. Here ‘\n’ means newline. It is an escape sequence in python, just like in C.

**Reading data from a file using loop**

f = open(‘sample’,’r’)

for line in f:

print(line)

f.close()

To read and write binary data file we need a **module** called **pickle**

**Error Handling in python**

There are 3 types of errors in computer programming.

**1. Syntax Errors**

These errors are due to the fact that the syntax of the language is not respected. The compiler can detect such errors. If syntax errors are present in the program then the compilation of the program fails and is terminated after showing the list of errors and the line number where the errors have occurred. In some cases the line number may not exactly indicate the correct place of the error. In some other cases, a single syntax error can result in a long list of errors. Correction of one or two errors in the program may remove the entire list of errors.

**2. Semantic Errors**

Errors due to an improper use of program statements.

**3. Logical Errors**

These errors are related to the logic of the program. Logical errors are also not detected by compiler and cause incorrect results. These errors occur due to incorrect translation of algorithm into the program, poor understanding of the problem and a lack of clarity of hierarchy of operators. From the point of view of when errors are detected, we can categorize these errors in the following types of errors.

**1.Compile Time Errors**

syntax errors and static semantic errors indicated by the compiler.

**2.Run Time Errors**

dynamic semantic errors, and logical errors, that cannot be detected by the compiler.

The first one is handled by the compiler/interpreter itself. Run time errors are the errors that occur during the execution of the program.

Some examples are, dividing by zero error, insufficient memory for dynamic memory allocation, referencing an out-of-range array element. These are not detected by compiler while compilation process. A program with these kinds of errors will run but produce erroneous results or may cause termination of program. Detection and removal of a run-time error is a difficult task.

Here we study how python handle the run time errors.

Errors detected during execution are called exceptions. Most exceptions are not handled by programs, however, and result in error messages while executing the program. Exceptions are different types and and the type is printed as part of the message.

**ZeroDivisionError:, NameError:, TypeError: are examples of exception types.**

**Exercises**

1. What is binary data? Explain with examples.

Study the module pickle and explain how to use this module for reading and writing a binary file.

2. Write a program to read msword.doc format file and.docx format file and examine the two formats and explain the differences.

3. What is an escape sequence? List out all escape sequences used in python and explain it.

4. Study the try and except statements and explain with examples.