5.3 PYTHON PROGRAMMING

L	P
3	4

RATIONALE

This subject introduces to the students the Python language. Upon completion of this subject, the student will be able to write non trivial Python programs dealing with a wide variety of subject matter domains. Topics include language components, the IDLE/IDE environment, control flow constructs, strings, I/O, collections, classes, modules, and regular expressions.

COURSE OUTCOMES

After undergoing the subject, the students will be able to:

CO1: Implement Python programs utilizing arithmetic expressions, repetition, file Inputand Output.

CO1: Demonstrate the use of the built-in data structures in Python.

CO3: Employ control structures, functions, and arrays to create Python programs.

CO4: Understand the concepts of object-oriented programming as used in Python.

CO5: Define the use of GUI and databases using Python.

DETAILED CONTENTS

UNIT I

The way of the program: The Python programming language, What is a program? What is debugging?, Syntax errors, Runtime errors, Semantic errors, Experimental debugging.

Variables, Expressions and Statements: Values and data types, Variables, Variable names and keywords, Statements, Evaluating expressions, Operators and operands, Type converter functions, Order of operations, Operations on strings, Input, Composition, The modulus operator.

Conditionals: Boolean values and expressions, Logical operators, Simplifying Boolean Expressions, Conditional execution, Chained conditionals, Nested conditionals, The return statement, Logical opposites.

UNIT II

Iteration: Assignment, Updating variables, The for loop, The while statement, The Collatz 3n + 1

sequence, Tables, Two-dimensional tables, Paired Data, Nested Loops for Nested Data.

Strings: Working with strings as single things, Working with the parts of a string, Length, Traversal and the for loop, Slices, String comparison, Strings are immutable, The in and not in operators, A find function, Looping and counting, Optional parameters, The built-in find method, The split method, Cleaning up your strings, The string format method.

Tuples: Tuples are used for grouping data, Tuple assignment, Tuples as return values, Composability of Data Structures.

Lists: List values, Accessing elements, List length, List membership, List operations, List slices, Lists are mutable, List deletion, Objects and references, Aliasing, Cloning lists, Lists and for loops, List parameters, List methods, Pure functions and modifiers, Functions that produce lists, Strings and lists, list and range, Nested lists, Matrices.

Functions: Functions with arguments and return values.

UNIT III

Modules: Random numbers, The time module, The math module, Creating your own modules, Namespaces, Scope and lookup rules, Attributes and the dot operator.

Files: About files, Writing our first file, Reading a file line-at-a-time, Turning a file into a list of lines, Reading the whole file at once, Working with binary files, Directories, fetching something from the web. **List Algorithms:** Linear search, Binary search, Merging two sorted lists.

UNIT IV

Object oriented programming: Classes and Objects- The Basics, Attributes, Adding methods to our class, Instances as arguments and parameters, Converting an instance to a string, Instances as return values, Objects are mutable, Sameness, Copying.

Exceptions: Catching exceptions, raising our own exceptions, the finally clause of the try statement

Inheritance: Polymorphism, Generalization, Pure functions.

UNIT V

GUI: Creating Graphical User Interfaces, Using Module Tkinter, Building a Basic GUI, Models, Views, and Controllers, Customizing the Visual Style, Few More Widgets.

Databases: Overview, Creating and Populating, Retrieving Data, Updating and Deleting, Using NULL for Missing Data, Using Joins to Combine Tables, Keys and Constraints, Advanced Features.

PRACTICAL EXERCISES

Part A

1. Let list1 and list2 be two lists of integers. Implement function sublist() that takes as input lists list1 and list2 and returns True if list1 is a sublist of list2, and False otherwise.

   >>> sublist([15, 1, 100], [20, 15, 30, 50, 1, 100])

   True

   >>> sublist([15, 50, 20], [20, 15, 30, 50, 1, 100])

   False

2. Write function vowelCount() that takes a string as input and counts and prints the numberof occurrences of vowels in the string.

   >>> vowelCount('Le Tour de France')

   a, e, i, o, and u appear, respectively, 1, 3, 0, 1, 1 times.

3. The cryptography function crypto() takes as input a string (i.e., the name of a file in the current directory). The function should print the file on the screen with this modification: Every occurrence of string 'secret' in the file should be replaced with string 'xxxxxx'.

   >>> crypto('crypto.txt')

   I will tell you my xxxxxx. But first, I have to explainwhy
   it is a xxxxxx.
   And that is all I will tell you about my xxxxxx.

4. Write a function stats() that takes one input argument: the name of a text file. The function should print, on the screen, the number of lines, words, and characters in the file;your function should open the file only once.

   >>>stats('example.txt')
   line count: 3
   word count: 20

   character count: 98

5. Implement function distribution () that takes as input the name of a file (as a string). This one-line file will contain letter grades separated by blanks. Your function should print the distribution of grades, as shown.

   >>> distribution('grades.txt')6
   students got A
   2 students got A-3

students got B+2
students got B 2
students got B-4
students got C 1
student got C- 2
students got F

6. The function censor () takes the name of a file (a string) as input. The function should open the file, read it, and then write it into file censored.txt with this modification: Every occurrence of a four-letter word in the file should be replaced with string 'xxxx'.
>>> censor ('example.txt')

Note that this function produces no output, but it does create file censored.txt in the current folder.

7. Create a dictionary for phones and their prices. Write functions to add a new entry (phone:price) ,search for a particular phone and retrieve it's price, given price findphones with same price , remove an entry, display all phones sorted according to price. [Program must be menu driven]
8. Write a Python program that prompts the user to enter a list of first names and stores them in a list. The program should display how many times the letter 'a' appears within the list.
9. Write a Python program that prompts the user to enter integer values for each of two lists.It then should displays whether the lists are of the same length, whether the elements in each list sum to the same value, and whether there are any values that occur in both lists.
10. Implement and test a Python program that determines if all parentheses in an entered line of code form matching pairs. Note: Pairs of parentheses may be nested.
11. Suppose variable s has been assigned in this way:

s = '"It was the best of times, it was the worst of times; it
was the age of wisdom, it was the age of foolishness; it was the
epoch of belief, it was the epoch of incredulity; it was ..."' Then
do the following, in order, each time:

(a) Write a sequence of statements that produce a copy of s, named newS, in whichcharacters ., ,, ;, and \n have been replaced by blank spaces.
(b) Remove leading and trailing blank spaces in newS (and name the new string newS).
(c) Make the all characters in newS lowercase (and name the new string newS).
(d) Compute the number of occurrences in newS of string 'it was'.

(e) Change every occurrence of was to is (and name the new string newS).

(f) Split newS into a list of words and name the list listS.

12. The function avgavg() takes as input a list whose items are lists of three numbers. Each three-number list represents the three grades a particular student received for a course. For example, here is an input list for a class of four students:
[[95,92,86], [66,75,54],[89, 72,100],[34,0,0]]

The function avgavg() should print, on the screen, two lines. The first line will contain a list containing every student's average grade. The second line will contain just one number: the average class grade, defined as the average of all student average grades.
>>> avgavg([[95, 92, 86], [66, 75, 54],[89, 72, 100], [34, 0, 0]])

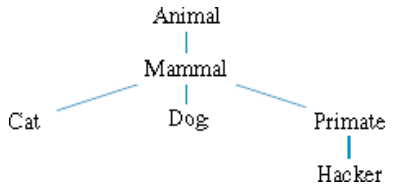
[91.0, 65.0, 87.0, 11.333333333333334]

63.5833333333

13. Implement function names () that takes no input and repeatedly asks the user to enter the first name of a student in a class. When the user enters the empty string, the function should print for every name the number of students with that name.
>>> names ()

Enter next name: Valerie
Enter next name: Bob Enter
next name: ValerieEnter
next name: AmeliaEnter
next name: Bob Enter next
name:
There is 1 student named Amelia
There are 2 students named Bob
There are 2 students named Valerie

14. Consider the class tree hierarchy:



Implement six classes to model this taxonomy with Python inheritance. In class Animal, implement method speak() that will be inherited by the descendant classes of Animal as is.

Complete the implementation of the six classes so they exhibit this behavior:

```
>>> garfield = Cat()
>>> garfield.speak()
Meeow
>>> dude = Hacker()
>>> dude.speak( )
Hello world!
```

Part B

1. Numerologists claim to be able to determine a person's character traits based on the numeric value" of a name. The value of a name is determined by summing up the values of the letters of the name where 'a' is 1, 'b' is 2, 'c' is 3 etc., up to 'z' being 26. For example,the name ""“Zelle""”would have the value 26+5+12+12+5=60 (which happens to be a very auspicious number, by the way). **Write a program that calculates the numeric value of a single name provided as input.** (Hint: Use dictionary, strings and its methods)
2. Expand your solution to the previous problem to allow the **calculation of a complete name** such as “John Marvin Zelle”” or “”John Jacob Jingleheimer Smith”. The total value is just the sum of the numeric values of all the names.
3. **Write a python program with function inner_product(x,y)** that computes the inner product of two (same length) lists. For example: list1=[1,2,3,4,5] and list2=[1,2,3,4,5]. The inner product list is inner_product=[1,4,9,16,25].
4. The **Sieve of Eratosthenes** is an elegant **algorithm** for finding all of the prime numbers up to some limit n. The basic idea is to first create a list of numbers from 2 to n. The first number is removed from the list, and announced as a prime number, and all multiples of this number up to n are removed from the list. This process continues until the list is empty.
    a) For example, if we wished to find all the primes up to 10, the list would originally contain 2, 3, 4, 5, 6, 7, 8, 9, 10.
    b) The 2 is removed and announced to be prime.
    c) Then 4, 6, 8, and 10 are removed, since they are multiples of 2.
    d) That leaves 3, 5, 7, 9.
    e) Repeating the process, 3 is announced as prime and removed, and 9 isremoved because it is a multiple of 3.
    f) That leaves 5 and 7. The algorithm continues by announcing that 5 is prime and removing it from the list.

g) Finally, 7 is announced and removed, and we're done.

Write a program that prompts a user for n and then uses the sieve algorithm to find all the primesless than or equal to n. (Hint: Use list. Remove () method)

5. Write a function that returns the index of the smallest element in a list of integers. If the number of such elements is greater than 1, return the smallest index. Use the following header: def index Of Smallest Element (lst): Write a program that prompts the user to enter a list of numbers, invokes this function to return the index of the smallest element, and displays the index.

6. **(Count occurrences of numbers)** Write a program that reads an unspecified number of integers and finds the ones that have the most occurrences. For example, if you enter 2 3 40 3 5 4 –3 3 3 2 0, the number 3 occurs most often. Enter all numbers in one line. If not one but several numbers have the most occurrences, all of them should be reported. For example, since 9 and 3 appear twice in the list 9 30 3 9 3 2 4, both occurrences should be reported.

7. **Morse Code Encryption/Decryption Program:** Develop and test a Python program that allows a user to type in a message and have it converted into Morse code, and also enter Morse code and have it converted back to the original message. The encoding of Morse code is given below:

Format the original message (containing English words) so that there is one sentence per line.

A	.-	N	-.
B	-...	O	---
C	-.-.	P	.--.
D	-..	Q	--.-
E	.	R	.-.
F	..-.	S	...
G	--.	T	-
H	....	U	..-
I	..	V	...-
J	.---	W	.--
K	-.-	X	-..-
L	.-..	Y	-.--
M	--	Z	--..

Format the Morse code fi le (containing dots and dashes) so that there is one letter per line,

with a blank line following the last letter of each word, and two blank lines following the end of each sentence (except the last).

RECOMMENDED BOOKS

1. A Downey, J. Elkner, and C. Meyers, “How to think like a computer scientist: learning with python. Green Tea Press”, Wellesley, Massachusetts, 2002.
2. J. Campbell, P. Gries, J. Montojo, and G. Wilson, “Practical programming: an introduction to computer science using Python”, Pragmatic Bookshelf, Second Edition, 2013.
3. A. B. Downey, “Python for software design: how to think like a computer scientist”, Cambridge University Press, 2009.
4. Z. A. Shaw, “Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code”, Addison-Wesley, 2013.

RECOMMENDED WEBSITES

1. http://swayam.gov.in

INSTRUCTIONAL STRATEGY

This is hands on practice based subject and topics taught in the class should be practiced in the Lab regularly for development of required skills among the students. This subject contains five units of equal weightage.