Software Engineering Lab

Spring 2024

Lab-3: Basics of Programming in Python

P1: Input-output in Python

1. Write a program to read the name of the user and greet him with a hello message.

2. See the following program. The objective of the program is to read two integer values from the console and print the sum of the numbers. If the following program does not work, do the needful correction.

```
num1 = input("Enter num1: ")  #Read the first number
num2 = input("Enter num2: ")  #Read the second number

num3 = num1 + num2            #Add two numbers
print("Product is: ", num3)
```

Hint: The return type of input() function is string, so we need to convert the input to integer as int(input())

3. Write a program which will read an integer numbers and print the numbers just before and after the number.

4. You have to print the following message in nicely formatted way as given below.

The Quick Brown Fox jumps over the Lazy Dog
and
Pack my box with Five Dozen Liquor Jugs

5. In the following program, see the last two print statements. Run the program and comment on if they work for your or not.

```
# This program should read first name and last name
# of the user and display the name that user has entered.

fName = input("Enter your first name: ")
print("\nThanks")
lName = input("Enter your last name: ")

print("\n" + fName + " " + lName)

print(fName, lName)
```

6. The following program gives an idea about more on ‘print’ statement in Python. The use of sep=’ ‘ allows to separate the text with no space, blank space and or with any character. Also, the end=’ ‘ can be used to end a message with any character including ‘\n’ explicitly.

```
print('F','U', sep='', end='')    #Here default ’\n’ is suppressed
print('N')

#\n provides new line after printing the year
print('25',    '12',    '1997', sep='-',     end='\n')  #Extra space does not have any effect

#The different parts areseparated with comma and a special ‘@’ after the end instead of
default ‘\n’
print('Red','Green','Blue', sep=',', end='@')

print('Debasis') #print statement is delimited with a default '\n'
print("Debasis") #print statement is delimited with a default '\n'
```

7. The following program with string modulo operator (%), with which you can print with a number of format to print numbers.

```
# Python program showing how to use string modulo operator(%)

print("Two digit value : %2d, Float value : %5.2f" % (1, 354.1732))

print("Total students : %3d, Boys : %2d" % (240, 120))   # print
integer value

print("Octal of %2d is %7.3o" % (25, 25))   # print octal value

print("Gravitational constant, G = %10.3E" % (6.6743e-11))   # print
exponential value
```

8. A few more formatting you can learn from the following python program. Run and understand how they work.

```
print('I love {} for "{}!"'.format('Python', 'Programming', ‘for’, ‘Data Analytics’))

# Using format() method and referring a position of the object
print('{0} and {1}'.format('Debasis', 'Samanta'))

print('{1} and {0}'.format('Debasis', 'Samanta'))

print(f"I love {'Programming'} in \"{'Python'}!\"")   #Note the use of ‘f’ here

# Using format() method and referring a position of the object
print(f"{'Python'} is easy for {'programming'}")
```

P2: Data Types in Python

9. Write a program which will read two strings and then print them in one line.

10. Write a program to read a string and then print the length of characters in it.

11. Write a program which will read three words from the user and add them in a list, initially empty . Delete the last word from the list. Can you delete the first words from the list? Check and try.

12. Write a program which will merge the words “Welcome”, “to”, “Python”, “Programming” into a single string “Welcome to Python Programming”

13. Write a Python program to delete an i-th letter from anstring. Read the string and i from the user. Print the string after the removal. Hint: Use string slicing. Delete the previous string. Hint: Use del command.

14. A tuple of a student is (<name>, <marks>, <deptt>). For example, (‘Debasis’, 98.5, “CS”) is a tuple. Create a list which contains 3 different tuples in it. Print the list.

15. The following is a program which shows how to create a set given a list and can be updated. Consider another list, say list4 = [1, 3, 5, 7, 9, 11, 13] and update the list list3 with the elements from the list list4.

```
# Python program to demonstrate the use of update() method

list1 = [1, 2, 3]
list2 = [5, 6, 7]
list3 = [10, 11, 12]

# Lists converted to sets
set1 = set(list2)
set2 = set(list1)

# Update method
set1.update(set2)

# Print the updated set
print(set1)

# List is passed as an parameter which gets automatically converted to a
set
set1.update(list3)
print(set1)
```

16. Write a program which will take the following lists and set to create a larger list, which should include all the elements in the list. Also, find the union, intersection and difference of the sets obtained from the list List1 and List2. Print all the sets you have obtained.

```
List1 = [1, 2, 3, 4]
List2 = [1, 4, 2, 3, 5]

Set3 = {'a', 'b', 'c'}
```

17. Given a set and a dictionary as below. Write a program which will include all the elements from the dictionary to set.

```
Set1 = {1, 2, 3, 4, 5}
myDict = {6: 'Six', 7: 'Seven', 8: 'Eight', 9: 'Nine', 10: 'Ten'}
```

Hint:

```
Set1.update(myDict)
```

18. The following program create a dictionary and then updating elemnets/ key values in it. Run the program and do the following tasks.
    (a) Add another element ‘Pine apple’ in the dictionary
    (b) Delete the element ‘Banana’

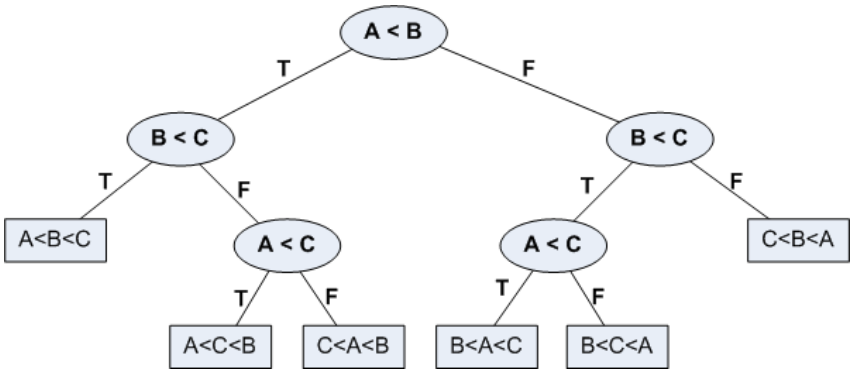
(c) Delete the element with key value 3

```
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)

# Adding elements one at a time
Dict[0] = 'Apple'
Dict[1] = 'Banana'
Dict[2] = ‘Mango’
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Adding set of values to a single Key
Dict[3] = ‘Orange’, ‘Cherry’, ‘Guava’
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Updating existing Key's Value
Dict[2] = 'Water Mellon'
print("\nUpdated key value: ")
print(Dict)
```

P3: Flow Control in Python

19. Write a program which should read any three integer numbers and print the largest number among them.

20. Hint: Use the following logic to write the program with nested if and if-else statements as appropriate.



21. Rewrite the program you have developed in 20 to print three numbers in a) ascending and b) descending order.

22. The following program does not work. Identify the error in the program, correct it and then run the program for the right output.

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

23. Generate n random numbers and print whether the sum of all the prime numbers is a prime.

P4: Functions in Python

24. Write a function to calculate the factorial value of any integer number n. Call the function for the different values of n

25. Write a function which will take two integers as arguments and return the average value of them.

26. Write a function which will take any two values of any type and swap the values.

27. Write a function which will take any three integer values as the arguments and return a list in ascending order.

28. Write a recursive function to calculate the following:
    (a) Factorial value of a positive integer passed as an argument.
    (b) Greatest common divisor (GCD) of two positive integer values.
    (c) The n-th Fibonacci number.
    (d) The sum of first n-harmonic numbers in a harmonic series.
    (e) Searching an element in a list of elements.

29. Write a function to return the number of characters in a string. The string value should be passed as an argument to the function.

30. Write a program which will read a character and then it will print as a consonant or vowel.

31. Two numbers are called co-prime numbers (also called relatively prime numbers) if those numbers have their greatest common divisor is 1. For example, (5, 9) are co-prime numbers; whereas (15, 20) are not co-primes. Write a program which will read two numbers from the user and check if they are co-prime numbers.

32. Write a function to read a list of numbers, store them in a list and use the list to find the following. Write iterative and recursive functions.

(a) Sum of the numbers
(b) Mean of the number
(c) Rang of the numbers
(d) Standard deviation of the numbers

33. Consider the following two programs. Check what sort of parameter passing are occurring in them.

Program 1:

```
def function1(x: int) -> int:
    print(”Inside the function: The initial value of x =%d”, x)
    x +=5
    print(”Inside the function: The value of x after the operation is %d”, x)

#Driver code
x = 0
print int(input(“Enter a value for x: “))
print(”The value of x =%d you have entered”, x)
function1(x)          #Call the function with argument x
print(”The value of x =%d after function is called”, x)
```

Program 2:

```
student={'Archana':28,'krishna':25,'Ramesh':32,'vineeth':25}
def test(student):
   new={'alok':30,'Nevadan':28}
   student.update(new)
   print("Inside the function",student)
   return
test(student)
print("outside the function:",student)
```

Hint:

If you pass immutable arguments like integers, strings or tuples to a function, the passing acts like **Call-by-value**. Program 1 is the example.

On the other hand, if we pass mutable arguments, such as lists, dictionaries, sets, etc. then the parameter passing is **Call-by-reference**. Program 2 is an example.

34. Write a Python function to count the occurrences of each word in a given sentence.

35. Create a function that returns the sum of digits in a given integer.

36. Write a Python function to find the common elements between two lists.

37. Write a function to reverse a string without using slicing.

38. Develop a function to check if a given number, is a palindrome.

39. Create a function that checks if a given string is a valid email address.

40. Write a function that will print any non-max number in a list.

P5: File handling in Python

Concept:

The key function for working with files in Python is the open() function.

fName = open(filename, fMode)

The open() function takes two parameters; filename, and mode.

There are four different modes for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist
"a" - Append - Opens a file for appending, creates the file if it does not exist
"w" - Write - Opens a file for writing, creates the file if it does not exist
"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode
"b" - Binary - Binary mode (e.g. images)

41: Creating a file to write some text into it

```
# Python code to create a file
file = open('test.txt','w')  # Opening the file in write mode
file.write("This is the write command")
file.write("It allows us to write in a particular file")
file.close()   # Closing the file

# Alternative method: with-write()
# Python code to illustrate with() along with write()
with open("test2.txt", "w") as f:
    f.write("Hello World!!!")
```

42: Opening an existing file

```
# a file named "test.txt" is already available in your machine
# This program will open the file in reading mode.

file1 = open('test.txt')      # Default is read and text mode
file2 = open('test.txt', 'r')
file3 = open('test.txt', 'rt')
```

```
# This will print every line one by one in the file
for each in file1:
    print (each)

# Alternative Way 1: Using read() method
# Python code to illustrate read() mode
file = open("text.txt", "r")
print (file.read())

# Alternative Way 3: with-read()
# Python code to illustrate with()
with open("text2.txt") as file:
    data = file.read()

print(data)

# Another Way 4: Reading a set of character of a given size
# Python code to illustrate read() mode character wise
file = open("text.txt", "r")
print (file.read(5))    # Read five characters, by default whole

#Alternative Way 5: Read all words in a file, using split() method
# Python code to illustrate split() function
with open("test.txt", "r") as file:
    data = file.readlines()
    for line in data:
        word = line.split()
        print (word)
```

43: Opening a file in append mode

```
# Python code to illustrate append() mode
file = open('test.txt', 'a')
file.write("This will add at the end of the file.")
file.close()

file1 = open('test.txt')
for each in file1:
    print (each)
```

44: Exercising the important file handling functions

```
import os

def create_file(filename):
    try:
```

```
        with open(filename, 'w') as f:
            f.write('Hello, world!\n')
        print("File " + filename + " created successfully.")
    except IOError:
        print("Error: could not create file " + filename)

def read_file(filename):
    try:
        with open(filename, 'r') as f:
            contents = f.read()
            print(contents)
    except IOError:
        print("Error: could not read file " + filename)

def append_file(filename, text):
    try:
        with open(filename, 'a') as f:
            f.write(text)
        print("Text appended to file " + filename + " successfully.")
    except IOError:
        print("Error: could not append to file " + filename)

def rename_file(filename, new_filename):
    try:
        os.rename(filename, new_filename)
        print("File " + filename + " renamed to " + new_filename + "
successfully.")
    except IOError:
        print("Error: could not rename file " + filename)

def delete_file(filename):
    try:
        os.remove(filename)
        print("File " + filename + " deleted successfully.")
    except IOError:
        print("Error: could not delete file " + filename)


if __name__ == '__main__':
    filename = "example.txt"
    new_filename = "new_example.txt"

    create_file(filename)
    read_file(filename)
```

```
    append_file(filename, "This is some additional text.\n")
    read_file(filename)
    rename_file(filename, new_filename)
    read_file(new_filename)
    delete_file(new_filename)
```

Python has a set of methods available for the file object.

Method	Description
close()	Closes the file
detach()	Returns the separated raw stream from the buffer
fileno()	Returns a number that represents the stream, from the operating system's perspective
flush()	Flushes the internal buffer
isatty()	Returns whether the file stream is interactive or not
read()	Returns the file content
readable()	Returns whether the file stream can be read or not
readline()	Returns one line from the file
readlines()	Returns a list of lines from the file
seek()	Change the file position
seekable()	Returns whether the file allows us to change the file position
tell()	Returns the current file position
truncate()	Resizes the file to a specified size
writable()	Returns whether the file can be written to or not
write()	Writes the specified string to the file
writelines()	Writes a list of strings to the file

45. Create a program that counts the number of vowels in a text file.

46. Write a Python script to remove duplicates elements in a file.

47. Write a Python program that reads the contents of a text file named "data.txt" and prints them.

48. Write a Python script to count the number of words in a text file.

49. Implement a program that searches for a specific word in a text file and prints its occurrences.

50. Write a program which will invert a file. Don’t use any extra file other than the input file for this.