Third Year B.C.A. (Under Science) Semester VI

Course BCA606 **Title: Python Lab Cource II**

Total Contact Hours: 48 hrs. **Total Credits: 04**

Total Marks: 100

Python Assignments:

1. Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.
2. Write a program to check whether the number is even or odd, print out an appropriate message to the user.
3. Write a program which will find all such numbers which are divisible by 7.
4. Write a program which can compute the factorial of a given numbers.
5. Write a program that prints out all the elements of the list that are less than 10.
6. Write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.
7. Write a program which accepts a sequence of comma-separated numbers from console and generate a list and a tuple which contains every number.Given the input: 34,67,55,33,12,98

   Then, the output should be:

   ```
   ['34', '67', '55', '33', '12', '98']
   ('34', '67', '55', '33', '12', '98')
   ```

8. Make a two-player Rock-Paper-Scissors game. (*Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game*) Rules:

   Rock beats scissors

   Scissors beats paper

   Paper beats rock
9. To determine whether the number is prime or not.
10. To check whether a number is palindrome or not. (using recursion and without recursion).
11. Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.Write two different functions to do this - one using a loop and constructing a list, and another using sets.
12. Write a program that asks the user how many Fibonnaci numbers to generate and then generates them.
13. Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order. E.g " I am tybca student" is :"student tybca am I"
14. Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password.

15. Write a program to implement binary search to search the given element using function.
16. Given a `.txt` file that has a list of a bunch of names, count how many of each name there are in the file, and print out the results to the screen.
17. Implement a function that takes as input three variables, and returns the largest of the three.(do not use max function)
18. Create a dictionary (in your file) of names and birthdays. When you run your program it should ask the user to enter a name, and return the birthday of that person back to them.
19. Write a program that takes a list of numbers (for example, `a = [5, 10, 15, 20, 25]`) and makes a new list of only the first and last elements of the given list.
20. Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.
21. Write a program that accepts a sentence and calculate the number of letters and digits.
22. Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters.

String:

A string is a sequence of characters. The string is a sequence of Unicode character in Python. Unicode was introduced to include every character in all languages and bring uniformity in encoding.

Strings can be created by enclosing characters inside a single quote or double quotes. Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

```
# All of the following are
equivalent my_string = 'Hello'
print(my_string) my_string =
"Hello"
print(my_string)     my_string
= '''Hello''' print(my_string)
# triple quotes string can extend multiple lines
my_string = """Hello, welcome
to           the world of Python"""
print(my_string)
```

The output of *stringm.py* will be:

```
Hello
Hello
Hello
Hello, welcome to
the world of Python
```

To access characters in a string:

We can access individual characters using indexing and a range of characters using slicing.
Index starts from 0. Trying to access a character out of index range will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError. Python allows negative indexing for its sequences.The index of -1 refers to the last item, -2 to the second last item and so on. We can access a range of items in a string by using the slicing operator (colon).

```
str = 'programing'
print('str = ', str)
#first character
print('str[0] = ',
str[0]) #last
character
print('str[-1] = ', str[-1])
#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])
#slicing 6th to 2nd last
character print('str[5:-2] = ',
str[5:-2])
```

Update string:
The existing string can be update by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether. For example
−

```
var1 = 'Hello World!' print "Updated
String :- ", var1[:6] + 'Python'
```

output:

```
Updated String :-  Hello Python
```

Python includes the following built-in methods to manipulate strings −

Sr.No.	Methods with Description
1	**capitalize()** Capitalizes first letter of string
2	**center(width, fillchar)** Returns a space-padded string with the original string centered to a total of width columns.
3	**count(str, beg= 0,end=len(string))** Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
4	**decode(encoding='UTF-8',errors='strict')** Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.

5	**encode(encoding='UTF-8',errors='strict')** Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	**endswith(suffix, beg=0, end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.
7	**expandtabs(tabsize=8)** Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	**find(str, beg=0 end=len(string))** Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.
9	**index(str, beg=0, end=len(string))** Same as find(), but raises an exception if str not found.
10	**isalnum()** Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
11	**isalpha()** Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	**isdigit()** Returns true if string contains only digits and false otherwise.
13	**islower()** Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	**isnumeric()** Returns true if a unicode string contains only numeric characters and false otherwise.
15	**isspace()** Returns true if string contains only whitespace characters and false otherwise.

16	**istitle()** Returns true if string is properly "titlecased" and false otherwise.
17	**isupper()** Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.
18	**join(seq)** Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	**len(string)** Returns the length of the string
20	**ljust(width[, fillchar])** Returns a space-padded string with the original string left-justified to a total of width columns.
21	**lower()** Converts all uppercase letters in string to lowercase.
22	**lstrip()** Removes all leading whitespace in string.
23	**maketrans()** Returns a translation table to be used in translate function.
24	**max(str)** Returns the max alphabetical character from the string str.
25	**min(str)** Returns the min alphabetical character from the string str.
26	**replace(old, new [, max])** Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	**rfind(str, beg=0,end=len(string))** Same as find(), but search backwards in string.
28	**rindex(str, beg=0, end=len(string))** Same as index(), but search backwards in string.

29	**rjust(width,[, fillchar])** Returns a space-padded string with the original string right-justified to a total of width columns.
30	**rstrip()** Removes all trailing whitespace of string.
31	**split(str="", num=string.count(str))** Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	**splitlines(num=string.count('\n'))** Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.
33	**startswith(str, beg=0,end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
34	**strip([chars])** Performs both lstrip() and rstrip() on string.
35	**swapcase()** Inverts case for all letters in string.
36	**title()** Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	**translate(table, deletechars="")**
	Translates string according to translation table str(256 chars), removing those in the del string.
38	**upper()** Converts lowercase letters in string to uppercase.
39	**zfill (width)** Returns original string leftpadded with zeros to a total of width characters; intended for numbers, zfill() retains any sign given (less one zero).

40	**isdecimal()** Returns true if a unicode string contains only decimal characters and false otherwise.

1. Write a Python function to calculate the factorial of a number (a non-negative integer). The function accepts the number as an argument.
2. Write a Python program to converting an Integer to a string in any base.
3. Write a Python program of recursion list sum.
4. Write a Python program to solve the Fibonacci sequence using recursion.
5. Write a Python program to get the sum of a non-negative integer.
6. Write a Python program to calculate the value of 'a' to the power 'b'
7. Write a Python program to find the greatest common divisor (gcd) of two integers
8. Write a Python function that takes a list and returns a new list with unique elements of the first list.
9. Write a Python function to check whether a number is perfect or not 10. Write a Python program to read a file line by line store it into an array.
11. Write a Python program to count the number of lines in a text file.
12. Write a Python program to count the frequency of words in a file.
13. Write a Python program to copy the contents of a file to another file 14. Write a Python program to read a random line from a file 15. Write a Python class to implement pow(x, n).
16. Write a Python class to reverse a string word by word.
    Input string : 'hello .py'
    Expected Output : '.py hello'
17. Write a Python class named Rectangle constructed by a length and width and a method which will compute the area and perimeter of a rectangle. –
18. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle