**MCAC0811: Python Programming Lab Program List**

**Module I**

1. **Introduction:**

**Compulsory**

* 1. Python Program to Print Hello world!
  2. Python Program to Add Two Numbers
  3. Python Program to Find the Square Root
  4. Python Program to Calculate the Area of a Triangle
  5. Python Program to Solve Quadratic Equation
  6. Python Program to Swap Two Variables
  7. Python Program to Generate a Random Number
  8. Python Program to Convert Kilometers to Miles
  9. Python Program to Convert Celsius To Fahrenheit

**Additional**

**Problem a. - Cafe menu**

Write a program that first displays a simple cafe menu (see example below), asks the user to enter the number of a choice, and either prints the appropriate action OR prints an error message that their choice was not valid.

Example output:

1. Soup and salad

2. Pasta with meat sauce

3. Chef's special

Which number would you like to order? 2

One Pasta with meat sauce coming right up!

Another example output:

1. Soup and salad

2. Pasta with meat sauce

3. Chef's special

Which number would you like to order? 5

Sorry, that is not a valid choice.

**Problem b. – Clock Time**

You look at the clock and it is exactly 2pm. You set an alarm to go off in 500 hours. At what time does the alarm go off?

1. **Decision Making and Loop**

**Compulsory**

* 1. Python program to check if a number is positive, negative or zero
  2. Python Program to Check if a Number is Odd or Even
  3. Python Program to Check Leap Year
  4. Python Program to Find the Largest Among Three Numbers
  5. Python Program to Check Prime Number
  6. Python Program to Print all Prime Numbers in an Interval
  7. Python Program to Find the Factorial of a Number
  8. Python Program to Display the multiplication Table
  9. Python Program to Print the Fibonacci sequence
  10. Python Program to Check Armstrong Number
  11. Python Program to Find Armstrong Number in an Interval
  12. Python Program to Find the Sum of Natural Numbers

**Additional**

**Problem 1 - Login security**

One important aspect of security in computer science is the concept of hashing: taking some text, and somehow converting it to a number. This is needed because many security algorithms work through math, so numbers are needed.

Another important aspect is the use of the modulo operator (%). You've seen this -- it returns the remainder portion of a division. This is useful because unlike most other math operators, modulo is one-way. That is, I can tell you that I'm thinking of a number x, and when I mod it by 5, I get 3, but from this information alone, you don't know whether x is 3 or 8 or 13 or 18, or ...

In this problem, we'll create a login screen, where the user must enter a password in order to see a secret message. We will give the user 3 chances to get the password right, and either print the secret message or a failure message (after 3 chances). First, define a function encrypt that takes one string. It will hash the string using the built-in Python function hash (try it on the shell) and modulo the value by a prime number (e.g. 541 -- this is very small in the computer science world but good enough for us). The function should then return this number. e.g. encrypt("mypassword") -> 283 (if you use 541 as the prime, for example)

At the top of the file, define a variable \_KEY to be the result, e.g. \_KEY = 283. Now, write the rest of the program. Each time you ask the user for the password, call encrypt with the user's input, and compare the value to \_KEY. If the two match, the user (most likely) entered the correct password, otherwise he loses one chance.

**Problem 2 - The game of Nims / Stones**

In this game, two players sit in front of a pile of 100 stones. They take turns, each removing between 1 and 5 stones (assuming there are at least 5 stones left in the pile). The person who removes the last stone(s) wins.

Write a program to play this game. This may seem tricky, so break it down into parts. Like many programs, we have to use nested loops. In the outermost loop, we want to keep playing until we are out of stones. Inside that, we want to keep alternating players. You have the option of either writing two blocks of code, or keeping a variable that tracks the current player. The second way is slightly trickier, by using the list.

Finally, we might want to have an innermost loop that checks if the user's input is valid. Is it a number? Is it a valid number (e.g. between 1 and 5)? Are there enough stones in the pile to take off this many? If any of these answers are no, we should tell the user and re-ask them the question.

So, the basic outline of the program should be something like this:

TOTAL = 100

MAX = 5

pile = TOTAL # all stones are in the pile to start

while [pile is not empty]:

while [player 1's answer is not valid]:

[ask player 1]

[check player 1's input... is it valid?]

[same as above for player 2]

Note how the important numbers 100 and 5 are stored in a single variable at the top. This is good practice -- it allows you to easily change the constants of a program. For example, for testing, you may want to start with only 15 or 20 stones.

Be careful with the validity checks. Specifically, we want to keep asking player 1 for their choice as long as their answer is not valid, BUT we want to make sure we ask them at least ONCE. So, for example, we will want to keep a variable that tracks whether their answer is valid, and set it to False initially. When you're finished, test each other's programs by playing them!

1. **String**

**Compulsory**

* 1. Python Program to Check Whether a String is Palindrome or not
  2. Python Program to Remove Punctuations From a String
  3. Python Program to Sort Words in Alphabetic Order
  4. Python Program to Count the Number of Each Vowel
  5. Count the number of characters (character frequency) in a string
  6. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.
  7. Get a single string from two given strings, separated by a space and swap the first two characters of each string

**Additional**

**Problem 1 – Collision detection of balls**

Many games have complex physics engines, and one major function of these engines is to figure out if two objects are colliding. Weirdly-shaped objects are often approximated as balls. In this problem, we will figure out if two balls are colliding.

We will think in 2D to simplify things, though 3D isn’t different conceptually. For calculating collision, we only care about a ball’s position in space and its size. We can store position with its center x-y coordinates, and we can use its radius for size. So a ball is a tuple of (x, y, r). To figure out if two balls are colliding, we need to compute the distance between their centers, then see if this distance is less than the sum of their radii. If so, they are colliding. Write a function that takes two balls and computes if they are colliding. Then call the function with two sets of balls. The first set is (0, 0, 1) and (3, 3, 1); these should not be colliding. The second set is (5, 5, 2) and (2, 8, 3); these should be colliding.

**Problem 2 – Pig-Latin Converter**

Write a program that lets the user enter in some English text, then converts the text to Pig-Latin. To review, Pig-Latin takes the first letter of a word, puts it at the end, and appends “ay”. The only exception is if the first letter is a vowel, in which case we keep it as it is and append “hay” to the end.

E.g. “hello”→ “ellohay”, and “image” → “imagehay”

It will be useful to define a list or tuple at the top called VOWELS. This way, you can check if a letter x is a vowel with the expression x in VOWELS.

It’s tricky for us to deal with punctuation and numbers with what we know so far, so instead, ask the user to enter only words and spaces. You can convert their input from a string to a list of strings by calling split on the string:

“My name is John Smith”.split(“ ”) → [“My”, “name”, “is”, “John”, “Smith”]

Using this list, you can go through each word and convert it to Pig-Latin. Also, to get a word except for the first letter, you can use word[1:].

Hints: It will make your life much easier – and your code much better – if you separate tasks into functions, e.g. have a function that converts one word to Pig-Latin rather than putting it into your main program code.

1. **Function**

**Compulsory**

* 1. Python Program To Display Powers of 2 Using Anonymous Function
  2. Python Program to Find Numbers Divisible by Another Number
  3. Python Program to Convert Decimal to Binary, Octal and Hexadecimal
  4. Python Program to Find ASCII Value of Character
  5. Python Program to Find HCF or GCD
  6. Python Program to Find LCM
  7. Python Program to Find Factors of Number
  8. Python Program to Make a Simple Calculator
  9. Python Program to Display Fibonacci Sequence Using Recursion
  10. Python Program to Find Sum of Natural Numbers Using Recursion
  11. Python Program to Find Factorial of Number Using Recursion
  12. Python Program to Convert Decimal to Binary Using Recursion

**Additional**

**Problem 1 - Sorting a List**

Write a program that asks the user to enter 10 (positive) numbers. The program should then print the numbers in sorted order, from biggest to smallest. To do this, first write a function that takes a list and finds the largest element. It then 1) deletes that element from the list and 2) returns that element. Hint: You will need to store two variables in this function: the biggest number you've seen so far (remember to initially set this to 0), and its position. Then iterate over the list, and for each element, check if it's bigger than the biggest number you've seen so far. If it is, change both variables (remember to change BOTH)!

So in your main program, you'll have to keep calling this function (in a loop) until the list is empty and keep printing the number that is returned.

**Problem 2 – Report card with GPA**

Write a program where the user can enter each of his grades, after which the program prints out a report card with GPA. Remember to ask the user how many classes he took. Example output is below.

How many classes did you take? 4

What was the name of this class? English

What was your grade? 94

REPORT CARD:

English - 94

Math – 96

Science – 91

Social Studies - 88

Overall GPA – 92.25

Hints: You’ll want to use a for loop, and you’ll probably want to keep two lists, one for names and one for grades. Remember, add to lists with append.

**Module -II**

1. **Tuple, Dictionary, Class, File Handling, Exception Handling**
   1. Python Program to Add a Key-Value Pair to the Dictionary
   2. Python Program to Concatenate Two Dictionaries Into One
   3. Python Program to Check if a Given Key Exists in a Dictionary or Not
   4. Python Program to Generate a Dictionary that Contains Numbers (between 1 and n) in the Form (x,x\*x).
   5. Python Program to Sum All the Items in a Dictionary
   6. Python Program to Multiply All the Items in a Dictionary
   7. Python Program to Read the Contents of a File
   8. Python Program to Count the Number of Words in a Text File
   9. Python Program to Count the Number of Lines in a Text File
   10. Python Program to Read a String from the User and Append it into a File
   11. Python Program to Count the Occurrences of a Word in a Text File
   12. Python Program to Copy the Contents of One File into Another
   13. Python Program to Read a Text File and Print all the Numbers Present in the Text File
   14. Python Program to Find the Area of a Rectangle Using Classes
   15. Python Program to Append, Delete and Display Elements of a List Using Classes
   16. Python Program to Create a Class and Compute the Area and the Perimeter of the Circle
   17. Python Program to Create a Class which Performs Basic Calculator Operations
   18. Python Program to Create a Class in which One Method Accepts a String from the User and Another Prints it
2. **Regular Expression**
   1. We have a [phone list](https://www.python-course.eu/simpsons_phone_book.txt) of the Simpsons, yes, the famous Simpsons from the American animated TV series. There are some people with the surname Neu. We are looking for a Neu, but we don't know the first name, we just know that it starts with a J. Let's write a Python script, which finds all the lines of the phone book, which contain a person with the described surname and a first name starting with J
   2. We have an imaginary phone list of the Simpsons in a list. Not all entries contain a phone number, but if a phone number exists it is the first part of an entry. Then follows separated by a blank a surname, which is followed by first names. Surname and first name are separated by a comma. The task is to rewrite this example in the following way:

Allison Neu 555-8396

C. Montgomery Burns

Lionel Putz 555-5299

Homer Jay Simpson 555-7334