Python

Lab Exercises

Day 1

1. Define a function max() that takes two numbers as arguments and returns the largest of them. Use the if-then-else construct available in Python. (It is true that Python has the max() function built in, but writing it yourself is nevertheless a good exercise.)
2. Define a function max_of_three() that takes three numbers as arguments and returns the largest of them.
3. Define a function that computes the *length* of a given list or string. (It is true that Python has the len() function built in, but writing it yourself is nevertheless a good exercise.)
4. Write a function that takes a character (i.e. a string of length 1) and returns True if it is a vowel, False otherwise.
5. Write a function translate() that will translate a text into *Robber's language*. That is, double every consonant and place an occurrence of "o" in between. For example, translate("this is fun")should return the string "tothohisos isos fofunon".
6. Define a function sum() and a function multiply() that sums and multiplies (respectively) all the numbers in a list of numbers. For example, sum([1, 2, 3, 4]) should return 10, and multiply([1, 2, 3, 4]) should return 24.
7. Define a function reverse() that computes the reversal of a string. For example, reverse("I am testing") should return the string "gnitset ma I".
8. Define a function is_palindrome() that recognizes palindromes (i.e. words that look the same written backwards). For example, is_palindrome("radar") should return True.
9. Write a function is_member() that takes a value (i.e. a number, string, etc) x and a list of values a, and returns Trueif x is a member of a, False otherwise. (Note that this is exactly what the in operator does, but for the sake of the exercise you should pretend Python did not have this operator.)
10. Define a function overlapping() that takes two lists and returns True if they have at least one member in common, False otherwise. You may use your is_member() function, or the in operator, but for the sake of the exercise, you should (also) write it using two nested for-loops.
11. Define a function generate_n_chars() that takes an integer n and a character c and returns a string, n characters long, consisting only of c:s. For example, generate_n_chars(5,"x") should return the string "xxxxx". (Python is unusual in that you can actually write an expression 5 * "x" that will evaluate to "xxxxx". For the sake of the exercise you should ignore that the problem can be solved in this manner.)
12. Define a *procedure* histogram() that takes a list of integers and prints a histogram to the screen. For example, histogram([4, 9, 7]) should print the following:

    ****
    ****
    ****

Day 2

13. The function max() from exercise 1) and the function max_of_three() from exercise 2) will only work for two and three numbers, respectively. But suppose we have a much larger number of numbers, or suppose we cannot tell in advance how many they are? Write a function max_in_list() that takes a *list* of numbers and returns the largest one.

14. Write a program that maps a list of words into a list of integers representing the lengths of the correponding words.

15. Write a function find_longest_word() that takes a list of words and returns the length of the longest one. Modify the same to do with lambda expression.

16. Write a function filter_long_words() that takes a list of words and an integer n and returns the list of words that are longer than n. Modify the same to do with lambda expression.

17. Write a version of a palindrome recognizer that also accepts phrase palindromes such as "Go hang a salami I'm a lasagna hog.", "Was it a rat I saw?", "Step on no pets", "Sit on a potato pan, Otis", "Lisa Bonet ate no basil", "Satan, oscillate my metallic sonatas", "I roamed under it as a tired nude Maori", "Rise to vote sir", or the exclamation "Dammit, I'm mad!". Note that punctuation, capitalization, and spacing are usually ignored.

18. A *pangram* is a sentence that contains all the letters of the English alphabet at least once, for example: *The quick brown fox jumps over the lazy dog*. Your task here is to write a function to check a sentence to see if it is a pangram or not.

19. "99 Bottles of Beer" is a traditional song in the United States and Canada. It is popular to sing on long trips, as it has a very repetitive format which is easy to memorize, and can take a long time to sing. The song's simple lyrics are as follows:

99 bottles of beer on the wall,
99 bottles of beer.
Take one down, pass it around,
98 bottles of beer on the wall.

The same verse is repeated, each time with one fewer bottle. The song is completed when the singer or singers reach zero. Your task here is write a Python program capable of generating all the verses of the song.

20. Represent a small bilingual lexicon as a Python dictionary in the following fashion {"merry":"god", "christmas":"jul", "and":"och", "happy":gott", "new":"nytt", "year":"år"} and use it to translate your Christmas cards from English into Swedish. That is, write a function translate() that takes a list of English words and returns a list of Swedish words.

21. Write a function char_freq() that takes a string and builds a frequency listing of the characters contained in it. Represent the frequency listing as a Python dictionary. Try it with something likechar_freq("abbabcbdbabdbdbababababcbcbab").

Day 3

22. In cryptography, a *Caesar cipher* is a very simple encryption techniques in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals. *ROT-13* ("rotate by 13 places") is a widely used example of a Caesar cipher where the shift is 13. In Python, the key for ROT-13 may be represented by means of the following dictionary:

key = {'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', 'n':'a', 'o':'b', 'p':'c', 'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k', 'y':'l', 'z':'m', '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', 'N':'A', 'O':'B', 'P':'C', 'Q':'D', 'R':'E', 'S':'F', 'T':'G', 'U':'H', 'V':'I', 'W':'J', 'X':'K', 'Y':'L', 'Z':'M'}

Implement an encoder/decoder of ROT-13. Once you're done, you will be able to read the following secret message: *Pnrfne pvcure? V zhpu cersre Pnrfne fnynq!*

23. Define a simple "spelling correction" function correct() that takes a string and sees to it that 1) two or more occurrences of the space character is compressed into one, and 2) inserts an extra space after a period if the period is directly followed by a letter. E.g. correct("This is very funny and cool.Indeed!") should return"This is very funny and cool. Indeed!" Tip: Use regular expressions!

24. The *third person singular* verb form in English is distinguished by the suffix *-s*, which is added to the stem of the infinitive form: *run -> runs*. A simple set of rules can be given as follows:

1. *If the verb ends in y, remove it and add ies*
2. *If the verb ends in o, ch, s, sh, x or z, add es*
3. *By default just add s*

Your task in this exercise is to define a function make_3sg_form () which given a verb in infinitive form returns its third person singular form. Test your function with words like *try*, *brush*, *run* and *fix*. Note however that the rules must be regarded as heuristic, in the sense that you must not expect them to work for all cases. Tip: Check out the string method endswith().

25. In English, the *present participle* is formed by adding the suffix *-ing* to the infinite form: *go -> going*. A simple set of heuristic rules can be given as follows:

1. *If the verb ends in e, drop the e and add ing (if not exception: be, see, flee, knee, etc.)*
2. *If the verb ends in ie, change ie to y and add ing*
3. *For words consisting of consonant-vowel-consonant, double the final letter before adding ing*
4. *By default just add ing*

Your task in this exercise is to define a function make_ing_form () which given a verb in infinitive form returns its present participle form. Test your function with words such as *lie*, *see*, *move* and *hug*. However, you must not expect such simple rules to work for all cases.

26. Create a module by name ***wordprocess*** out of Q.23 to Q.25

Day 4

27. Implement the Luhn checksum algorithm given below which checks if given credit card number is valid or not:

```
sum = 0.
for each digit of credit card number, starting from right,
   if the digit's index is even,
      add the digit to sum.
   else,
      double the digit's value.
      if the doubled value is less than 10,
         add the doubled value to sum.
      else,
         split the doubled value into its two digits.
         add the first digit to sum.
         add the second digit to sum.
If sum is divisible by 10 then valid
Else invalid
```

```
      4  4  0  8  0  4  1  2  7  4  3  6  9  8  5  3
Scale *2    *2    *2    *2    *2    *2    *2    *2
     ------------------------------------------------------------
      8  4  0  8  0  4  2  2 14  4  6  6 18  8 10  3
Sum 8 + 4 + 0 + 8 + 0 + 4 + 2 + 2 + 1+4 + 4 + 6 + 6 + 1+8 + 8 + 1+0 + 3
    = 70
70 is divisible by 10, therefore this card number is valid.
```

If 4408041254369873, the following figure shows the algorithm summing the latter number in detail. Notice how digits at even indexes are doubled and potentially split into two digits if they exceed 10 when doubled. For example, the number 7 at index 8 which is doubled to 14 which split to make 1+4.

28. According to Wikipedia, a *semordnilap* is a word or phrase that spells a *different* word or phrase backwards. ("Semordnilap" is itself "palindromes" spelled backwards.) Write a semordnilap recogniser that accepts a file name (pointing to a list of words) from the user and finds and prints all pairs of words that are semordnilaps to the screen. For example, if "stressed" and "desserts" is part of the word list, the the output should include the pair "stressed desserts". Note, by the way, that each pair by itself forms a palindrome!

29. A *hapax legomenon* (often abbreviated to *hapax*) is a word which occurs only once in either the written record of a language, the works of an author, or in a single text. Define a function that given the file name of a text will return all its hapaxes. Make sure your program ignores capitalization.

30. Implement a class *Employee* (-name (length>3 and <20), -id (unique), -basicSal (>=10000) , constructor, +setname, +setid, +setsalary, +getname, +getid, +getsalary, +display, +calcSal(): 1.1 times basicSal, destructor).

A. Throw exception objects (accordingly for out of range values) by creating *EmpError* exception class.
B. Creating the class *Manager* which inherits employee, overrides the calcSal() function by giving 20% of increment in salary.
C. Write a unit testing module for all the functions.
D. Write a main to interactively accept the details and build a list of Employee and Manager Objects.
E. Create tables EMPLOYEE and MANAGER. Read from the list and insert all the contents into a database with tables EMPLOYEE and MANAGER respectively having the same attributes as the class. Delete those rows which have salary less than 1000