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 "rövarspråket" (Swedish for "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 True if 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:
    ****
    ****
    ****
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.
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.

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:
20. 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.
21. 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.
22. 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 like char_freq("abbabcbdbabdbdbababcbcbab").
23. 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:
24. 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'}

25. Your task in this exercise is to implement an encoder/decoder of ROT-13. Once you're done, you will be able to read the following secret message:
26. Pnrfne pvcure? V zhpu cersre Pnrfne fnynq!
27. Note that since English has 26 characters, your ROT-13 program will be able to both encode and decode texts written in English.
28. 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!

29. 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:
    - If the verb ends in y, remove it and add ies
    - If the verb ends in o, ch, s, sh, x or z, add es
    - By default just add s
30. 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().
31. 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:
    - If the verb ends in e, drop the e and add ing (if not exception: be, see, flee, knee, etc.)
    - If the verb ends in ie, change ie to y and add ing
    - For words consisting of consonant-vowel-consonant, double the final letter before adding ing
    - By default just add ing
32. 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.
33. ## Higher order functions and list comprehensions
34. Using the higher order function reduce(), write a function max_in_list() that takes a list of numbers and returns the largest one. Then ask yourself: why define and call a new function, when I can just as well call the reduce() function directly?
35. Write a program that maps a list of words into a list of integers representing the lengths of the correponding words. Write it in three different ways: 1) using a for-loop, 2) using the higher order function map(), and 3) using list comprehensions.
36. Write a function find_longest_word() that takes a list of words and returns the length of the longest one. Use only higher order functions.
37. Using the higher order function filter(), define 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.
38. 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. Use the higher order function map() to write a function translate() that takes a list of English words and returns a list of Swedish words.
39. Implement the higher order functions map(), filter() and reduce(). (They are built-in but writing them yourself may be a good exercise.)
40. ## Simple exercises including I/O
41. Write a version of a palindrome recogniser that accepts a file name from the user, reads each line, and prints the line to the screen if it is a palindrome.

42. 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!
43. Write a procedure char_freq_table() that, when run in a terminal, accepts a file name from the user, builds a frequency listing of the characters contained in the file, and prints a sorted and nicely formatted character frequency table to the screen.
44. The International Civil Aviation Organization (ICAO) alphabet assigns code words to the letters of the English alphabet acrophonically (Alfa for A, Bravo for B, etc.) so that critical combinations of letters (and numbers) can be pronounced and understood by those who transmit and receive voice messages by radio or telephone regardless of their native language, especially when the safety of navigation or persons is essential. Here is a Python dictionary covering one version of the ICAO alphabet:
45. d = {'a':'alfa', 'b':'bravo', 'c':'charlie', 'd':'delta', 'e':'echo', 'f':'foxtrot',
    'g':'golf', 'h':'hotel', 'i':'india', 'j':'juliett', 'k':'kilo', 'l':'lima',
    'm':'mike', 'n':'november', 'o':'oscar', 'p':'papa', 'q':'quebec', 'r':'romeo',
    's':'sierra', 't':'tango', 'u':'uniform', 'v':'victor', 'w':'whiskey',
    'x':'x-ray', 'y':'yankee', 'z':'zulu'}

46. Your task in this exercise is to write a procedure speak_ICAO() able to translate any text (i.e. any string) into spoken ICAO words. You need to import at least two libraries: os and time. On a mac, you have access to the system TTS (Text-To-Speech) as follows: os.system('say ' + msg), where msg is the string to be spoken. (Under UNIX/Linux and Windows, something similar might exist.) Apart from the text to be spoken, your procedure also needs to accept two additional parameters: a float indicating the length of the pause between each spoken ICAO word, and a float indicating the length of the pause between each word spoken.
47. 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.
48. Write a program that given a text file will create a new text file in which all the lines from the original file are numbered from 1 to n (where n is the number of lines in the file).
49. Write a program that will calculate the average word length of a text stored in a file (i.e the sum of all the lengths of the word tokens in the text, divided by the number of word tokens).
50. Write a program able to play the "Guess the number"-game, where the number to be guessed is randomly chosen between 1 and 20. (Source:http://inventwithpython.com)
    This is how it should work when run in a terminal:
    >>> import guess_number
    Hello! What is your name?
    Torbjörn
    Well, Torbjörn, I am thinking of a number between 1 and 20.
    Take a guess.
    10

Your guess is too low.
Take a guess.
15
Your guess is too low.
Take a guess.
18
Good job, Torbjörn! You guessed my number in 3 guesses!

51. An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once; e.g., orchestra = carthorse, A decimal point = I'm a dot in place. Write a Python program that, when started 1) randomly picks a word w from given list of words, 2) randomly permutes w (thus creating an anagram of w), 3) presents the anagram to the user, and 4) enters an interactive loop in which the user is invited to guess the original word. It may be a good idea to work with (say) colour words only. The interaction with the program may look like so:
    >>> import anagram
    Colour word anagram: onwbr
    Guess the colour word!
    black
    Guess the colour word!
    brown
    Correct!
52. In a game of Lingo, there is a hidden word, five characters long. The object of the game is to find this word by guessing, and in return receive two kinds of clues: 1) the characters that are fully correct, with respect to identity as well as to position, and 2) the characters that are indeed present in the word, but which are placed in the wrong position. Write a program with which one can play Lingo. Use square brackets to mark characters correct in the sense of 1), and ordinary parentheses to mark characters correct in the sense of 2). Assuming, for example, that the program conceals the word "tiger", you should be able to interact with it in the following way:
    >>> import lingo
    snake
    Clue: snak(e)
    fiest
    Clue: f[i](e)s(t)
    times
    Clue: [t][i]m[e]s
    tiger
    Clue: [t][i][g][e][r]
53. **Somewhat harder exercises**
54. A sentence splitter is a program capable of splitting a text into sentences. The standard set of heuristics for sentence splitting includes (but isn't limited to) the following rules:
55. Sentence boundaries occur at one of "." (periods), "?" or "!", except that
    - Periods followed by whitespace followed by a lower case letter are not sentence boundaries.
    - Periods followed by a digit with no intervening whitespace are not sentence boundaries.

- Periods followed by whitespace and then an upper case letter, but preceded by any of a short list of titles are not sentence boundaries. Sample titles include Mr., Mrs., Dr., and so on.
- Periods internal to a sequence of letters with no adjacent whitespace are not sentence boundaries (for example, www.aptex.com, or e.g).
- Periods followed by certain kinds of punctuation (notably comma and more periods) are probably not sentence boundaries.

56. Your task here is to write a program that given the name of a text file is able to write its content with each sentence on a separate line. Test your program with the following short text: Mr. Smith bought cheapsite.com for 1.5 million dollars, i.e. he paid a lot for it. Did he mind? Adam Jones Jr. thinks he didn't. In any case, this isn't true... Well, with a probability of .9 it isn't. The result should be:
57. Mr. Smith bought cheapsite.com for 1.5 million dollars, i.e. he paid a lot for it.
    Did he mind?
    Adam Jones Jr. thinks he didn't.
    In any case, this isn't true...
    Well, with a probability of .9 it isn't.

58. An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once; e.g., orchestra = carthorse. Using the word list at http://www.puzzlers.org/pub/wordlists/unixdict.txt, write a program that finds the sets of words that share the same characters that contain the most words in them.
59. Your task in this exercise is as follows:
    - Generate a string with N opening brackets ("[") and N closing brackets ("]"), in some arbitrary order.
    - Determine whether the generated string is balanced; that is, whether it consists entirely of pairs of opening/closing brackets (in that order), none of which mis-nest.
60. Examples:
61. [] OK][NOT OK
    [][] OK][][NOT OK
    [[][]] OK []][[] NOT OK

62. A certain childrens game involves starting with a word in a particular category. Each participant in turn says a word, but that word must begin with the final letter of the previous word. Once a word has been given, it cannot be repeated. If an opponent cannot give a word in the category, they fall out of the game. For example, with "animals" as the category,
63. Child 1: dog
    Child 2: goldfish
    Child 1: hippopotamus
    Child 2: snake
    ...

64. Your task in this exercise is as follows: Take the following selection of 70 English Pokemon names (extracted from Wikipedia's list of Pokemon) and generate the/a sequence with the highest possible number of Pokemon names where the subsequent name starts with the final letter of the preceding name. No Pokemon name is to be repeated.