100 Python Exercises

Adapted from the [Python Exercises of Jeffrey Hu](https://github.com/zhiwehu/Python-programming-exercises). [Github: Jeffrey Hu](https://github.com/zhiwehu)

Adaptation by: [Joseph Anthony Debono](https://github.com/jadebono). [Email Joseph](joe@jadebono.com)

Question 1

Write a program which will find all such numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a comma-separated sequence on a single line.

Solution:

```py
def div_seven():
   x = []
   for i in range(2000,3201):
       if i % 7 == 0 and i % 5 != 0:
        x.append(str(i))
   print(",".join(x))


def test():
   for i in range(1, 10):
       print(i)


if __name__ == "__main__":
   div_seven()
```

Question 2

Write a program which can compute the factorial of a given numbers. Suppose the following input is supplied to the program:

8 Then, the output should be: 40320

Hints:

1. In case of input data being supplied to the question, take the input from the user using input();
1. Use a try-except block for input validation.

Solution:

```py
def test_int(n):
   try:
       int(n)
       return True
   except (ValueError, TypeError):
       return False


def get_val():
   user_input = input("Please input an integer. Note, any input that is not an integer will be discarded:\n")
   while (test_int(user_input) == False):
       user_input = input("Please input an integer. Note, any input that is not an integer will be discarded:\n")
   return int(user_input)


def fac(n):
   if n == 0:
       return 1
   else:
       return n * fac(n - 1)


if __name__ == "__main__":
   n = get_val()
   print(fac(n))
```

Question 3

With a given integral number n, write a program to generate an object that contains (i, i*i) such that i is an integral number between 1 and n (both included). and then the program should print the object. Suppose the following input is supplied to the program: 8 Then, the output should be: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function;
1. Use a try-except block for input validation.

Solution:

```py
# function to test that input is an integer
def test_int(n):
   try:
       int(n)
       return True
   except (ValueError, TypeError):
       return False


# function to get integer
def get_int():
   num = "x"
   while not test_int(num) or int(num) < 1:
       num = input("please enter an integer. Any other input will
be discarded:\t")
   return int(num)


# function to create dictionary
def create_dic(i):
   return {i : i * i for i in range(1, i+1)}


if __name__ == "__main__":
   x = get_int()
   y = create_dic(x)
   print(y)
```

Question 4

Write a program which takes a sequence of comma-separated numbers and generates an array and an object which contains every number. Suppose the following input is supplied to the program:
34,67,55,33,12,98

Then, the output should be: ['34', '67', '55', '33', '12', '98']
{'34':34, '67':67, '55':55, '33':33, '12':12, '98':98}

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function;
1. Add input validation.

Solution:

```py
# func to validate that CSV contains just ints and comma separators
def validate_csv(csv):
   for elem in csv:
       if not elem.isdigit() and elem not in {","}:
           return False
   return True


# func to create array
def create_array(csv):
   return csv.split(",")


# func to create dictionary
def create_dic(csv):
   csv_arr = csv.split(",")
   return {i: i for i in csv_arr}


# this func prompts the user to input a CSV, and uses a flag for the while loop
def inp_csv():
   x = False
```

```
    while not x:
        csv = input("Please input a sequence of integers separated
only by a comma:\n")
        if validate_csv(csv):
            x = True
            return csv
        else:
            print("Invalid input!")


if __name__ == "__main__":
    csv = inp_csv()
    arr = create_array(csv)
    dic = create_dic(csv)
    print(arr, dic, sep='\n')
```

Question 5

Define a class which has at least two methods:

1. getString: to get a string from console input.
1. printString: to print the string in upper case.

Also please include simple test function to test the class
methods.

Hints:

1. Use __init__ method to construct some parameters.

Solution:

```py
class Twup:
    def __init__(self):
        self.name = "My name is twup"
        self.my_str = "No input yet"

    def get_string(self):
        self.my_str = input("Please input a string:\t")

    def print_string(self):
```

```
        print(self.my_str)


def run_class():
  myTwup = Twup()
  print(myTwup.name)
  myTwup.print_string()
  myTwup.get_string()
  myTwup.print_string()


if __name__ == "__main__":
   run_class()
```

Question 6

Write a program that calculates and prints the value according to the given formula: Q = Square root of [(2 * C * D)/H]

Following are the fixed values of C and H:

1. C = 50;
1. H = 30;
1. D is the variable whose values should be input to your program in a comma-separated sequence, ex: 100,150,180.

Example:

Let us assume the following comma separated input sequence is given to the program: 100,150,180

The output of the program should be: 18,22,24

Hints:

1. If the input received is in decimal form, it should be rounded off to its nearest value (for example, if the input received is 26.0, it should be printed as 26);
1. In case of input data being supplied to the question, it should be assumed to come from the input() function.

```py
```

```
import math

# validation of the csv string - obviously needed only if the csv
comes from the user input
def validate_csv(csv):
   tokens = [i for i in csv.split(",")]
   for elem in tokens:
       # nested try-except block to test that tokens are either
integers or decimals
       try:
           int(elem)
       except Exception:
           try:
               float(elem)
           except Exception:
               return False
   return True


# decimals are rounded using banker's rounding. To round up with
0.5 use the decimal module
def process_arr(csv):
   ans = []
   d_arr = []
   C = 50
   H = 30
   for elem in csv.split(","):
       # conditional to convert elem to int if it is a float
       if float(elem):
           d_arr.append(int(round(float(elem))))
       else:
           d_arr.append(int(elem))
   ans = [round(math.sqrt((2 * C * elem)/H)) for elem in d_arr]
   return ",".join([str(i) for i in ans])


if __name__ == "__main__":
   d_csv = "100.3,150,180"
   if validate_csv(d_csv):
       new_csv_a = process_arr(d_csv)
       print(new_csv_a)
```

Question 7

Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array. The element value in the i-th row and j-th column of the array should be i*j. Note: i=0,1.., X-1; j=0,1,Y-1.

Example: Suppose the following inputs are given to the program: 3,5 Then, the output of the program should be: [[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]

Hints:

1. Create an array;
1. The external loop (rows) should create an empty array for each of its iterations and push each array into the parent array;
1. The inner loop should multiply the current value of row with each of the values of cols and push each product into the current iteration of array[rows];
1. In case of input data being supplied to the question, it should be assumed to come from the input() function.

Solution:

```py
# input validation
def input_validation(i, j):
   return True if (i.isdigit() and j.isdigit()) and (int(i) and int(j)) and (int(i) <= 10 and int(j) <= 10) else False


# input
def get_input():
   while True:
       i = input("please input no of rows:\t")
       j = input("please input no of columns:\t")
       if (not input_validation(i, j)):
           print("One or more of your inputs are invalid! Please try again.")
       else:
           break
   return int(i), int(j)


# array generator
```

```
def gen_arr(i, j):
   return [[i * j for j in range(0, j)] for i in range(0, i)]


if __name__ == "__main__":
   print("Please input integers to create a 2D array. For the
sake of this exercise, keep the numbers to a maximum of 10.")
   i, j = get_input()
   twod_arr = gen_arr(i, j)
   print(twod_arr)
```

Question 8

Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.
Suppose the following input is supplied to the program:
without,hello,bag,world
Then, the output should be: bag,hello,without,world

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function;
2. Add input validation to ensure that the supplied string is indeed a comma-separated sequence.

Solution:

```py
# input validation
def input_validation(inp):
   test = [i for i in inp]
   comma = False
   for i in test:
       if i == ",":
           comma = True
   return comma


# input
def get_input():
```

```
    flag = False
    while (not flag):
        inp = input("Please input a sequence of strings separated
by comma:\t")
        flag = input_validation(inp)
        if (not flag):
            print("This is not a comma-separated sequence! Please
try again.")
    return inp


if __name__ == "__main__":
    inp = [i for i in get_input().split(",")]
    # note: .sort() sorts the old array in place without returning
a new value. Use sorted to return a new sorted array
    inp.sort()
    print(",".join(inp))g
```

Question 9

Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized. Suppose the following input is supplied to the program:

```
Hello world
Practice makes perfect
```

Then, the output should be:

```
HELLO WORLD
PRACTICE MAKES PERFECT
```

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function;
1. Keep offering the user the faculty of inputting lines until the user inputs a "q";
1. There is no need for input validation, the program will ignore numbers and special characters;
1. Use the .upper() method.

Solution:

```py
# input function
def line_inputs():
   line_arr = []
   flag = False
   print("Please type in a line to capitalize. The program will
keep accepting lines until you input the letter q")
   while (not flag):
       inp = input("Please input a line to capitalize:\t")
       flag = True if inp == "q" else line_arr.append(inp)
   return line_arr


# The capitalize function
def capitalize_arr(inp_arr):
   for line in inp_arr:
       print(line.upper())


if __name__ == "__main__":
   inp_arr = line_inputs()
   capitalize_arr(inp_arr) if len(inp_arr) > 0 else print("you
have inputted no lines")
```

Question 10

Write a program that accepts a sequence of whitespace separated words in a single string as input and prints the words in ascending order after removing all duplicate words and sorting them alphanumerically. Output them once more as whitespace-separated string.

Suppose the following input is supplied to the program:
hello world and practice makes perfect and hello world again

Then, the output should be:
again and hello makes perfect practice world

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function.

Solution:

```py
# input validation
def input_validation(wsv):
   wsv_arr = [i for i in wsv.split(" ")]
   return True if len(wsv_arr) > 1 else False


# input
def get_input():
   wsv = input("Please input your whitespace-separated string here:\t")
   while True:
       if (not input_validation(wsv)):
           wsv = input("Your input did not consist of whitespace-separated values. Please try again:\t")
       else:
           break
   return wsv


# sort function
def sort_arr(wsv):
   #  no need to convert set to list again since you're converting it to a whitespace-separated string once more.
   return (" ").join(sorted(set([i for i in wsv.split(" ")])))


if __name__ == "__main__":
   print("Please input a string of words separated by whitespace. This program will remove all duplicate words and print them out in ascending order")
   wsv = get_input()
   sorted_wsv = sort_arr(wsv)
   print(sorted_wsv)
```

Question 11

Write a program which accepts a sequence of comma separated 4 digit binary numbers as its input and then check whether they are divisible by 5 or not. The numbers that are divisible by 5 are to be printed in a comma separated sequence.

Suppose the following input: 0100,0011,1010,1001 Then the output should be: 1010.

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function.

Solution:

```py
# input validation. This function must validate that there are commas in the sequence, that once the string is split, each element is
# in fact 4 characters long, and that each of the characters in each element corresponds to 1 or 0, meaning that it can resolve to a
# binary number.
def input_validation(csbs):
   if "," in list(csbs):
       bins_list = csbs.split(",")
       # test that each element in the bins_list is in fact 4 characters long
       for i in bins_list:
           if len(i) != 4:
               return False
       #  take every elem in bins_list and test that it is a binary
       for elem in bins_list:
           # check that every char in elem is to be found in "01"
           for i in elem:
               if i not in "01":
                   return False
   # if no comma is found in the string
   else:
       return False
   #  returns True if all requirements are satisfied
   return True
```

```
# input function
def get_binaries():
   csbs = input("Please input a sequence of comma-separated 4
digit binary numbers. No other input will be accepted:\t")
   while True:
       if (not input_validation(csbs)):
           csbs = input("Your input was invalid. Please try
again:\t")
       else:
           break
   return csbs.split(",")

# the binary division
def bin_div(bin_arr):
   csbs_arr = [i for i in bin_arr if int(i, 2) % 5 == 0]
   print(",".join(csbs_arr)) if len(csbs_arr) > 0 else
print("None of your supplied binary numbers is divisible by 5.")


if __name__ == "__main__":
   csbs = get_binaries()
   bin_div(csbs)
```

Question 12

Write a program, which will take two numbers (such as 1000 and 3000) and find all such numbers between those two numbers (both included) such that each digit of the number is an even number. The numbers obtained should be printed in a comma-separated sequence on a single line.

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function.

Solution:

```py
# input validation. This function must validate that the inputs
are integers
```

```
def input_validation(a, b):
   try:
       int(a) and int(b)
       return True
   except Exception:
       return False


# input function
def get_ints():
   print("For the following code, only integers will be excepted,
nothing else.")
   while True:
       a = input("Please input an integer:\t")
       b = input("Please input another integer:\t")
       if (not input_validation(a, b)):
           print("invalid inputs! Try again.")
       else:
           break
   n_arr = sorted([int(a), int(b)])
   return n_arr[0], n_arr[1]


# the digit analyser function
def is_even(a, b):
   return "".join([str(i) for i in range(a, b + 1) if
all(int(elem) % 2 == 0 for elem in str(i))])


if __name__ == "__main__":
   a, b = get_ints()
   csv_ints = is_even(a, b)
   print(csv_ints)
```

Question 13

Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters. Suppose the following input is supplied to the program: Hello world! Then, the output should be:

UPPER CASE 1

LOWER CASE 9

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function;
1. Input validation not needed for this one.

```py
def get_string():
   return input("Please input a sentence or sequence of characters. The program will count the number of letters and digits supplied:\t")


def count_letters_and_digits(input_str):
   letter_counter = 0
   digit_counter = 0
   for elem in input_str:
       if elem.isalpha():
           letter_counter += 1
       elif elem.isdigit():
           digit_counter += 1
   return f'The number of letters in your string is {letter_counter} while the number of digits is {digit_counter}.'


if __name__ == "__main__":
   my_str = get_string()
   get_results = count_letters_and_digits(my_str)
   print(get_results)
```

Question 14

Write a program that accepts a sentence and calculate the number of upper case letters and lower case letters. Suppose the following input is supplied to the program: Hello world! Then, the output should be:

1. UPPER CASE 1
1. LOWER CASE 9

Hints:

1. In case of input data being supplied to the question, it should be assumed to come from the input() function;
1. Use .isupper() and .islower() for this problem as only these two methods will restrict their counting to uppercase and lowercase letters;
1. Input validation not needed for this one.

```py
def get_string():
   return input("Please input a sentence or sequence of characters. The program will count the number of lowercase and uppercase letters in your input:\t")


def count_lower_and_upper(input_str):
   lower_counter = 0
   upper_counter = 0
   for elem in input_str:
       if elem.islower():
           lower_counter += 1
       elif elem.isupper():
           upper_counter += 1
   return f'The number of lowercase letters in your string is {lower_counter} while the number of uppercase ones is {upper_counter}.'


if __name__ == "__main__":
   my_str = get_string()
   get_results = count_lower_and_upper(my_str)
   print(get_results)

```

Question 15a

Write a program that computes the value of a+aa+aaa+aaaa with a given digit as the value of a. Suppose the following input is supplied to the program: 9 Then, the output should be: 11106.

Hints:

1. Only ask the user to supply the value to represent "a" in the sequence "a+aa+aaa+aaaa";
1. To be clear, the user must output the sum of "9 + 99 + 999 + 9999" given a value of 9 NOT "9 + (9*9) + (9*9*9) + (9*9*9*9)";
1. Add input validation to ensure that the supplied value is an integer.

```py
def validate_input(n):
   return True if n.isdigit() else False

def get_input():
   n = input("Please input an integer value for 'a' to calculate the value of a+aa+aaa+aaaa. Only integers will be accepted:\t")
   while True:
       if (not validate_input(n)):
           n = input("You did not input a integer. Please do so:\t")
       else:
           break
   return int(n)

def calculate_value(n):
   a_counter = 0
   for elem in "a+aa+aaa+aaaa".split("+"):
       # This line first turns an iteration of n into a string for every number in the range of the length of elem
       # Then it puts that str(n) into a list, which is then joined into a single string and finally turned into an integer that can
       # be added to a_counter
       a_counter += int("".join([str(n) for i in range(len(elem))]))
   print(a_counter)


if __name__ == "__main__":
   n = get_input()
   calculate_value(n)
```

Question 15b

Write a program that computes the value of a+aa+aaa+aaaa with a given digit as the value of a. Each cluster of a is an exponent of a multiplied by itself for the number of iterations in its sequence. Suppose the following input is supplied to the program: 9 Then, the output should be: 11106.

Hints:

1. Only ask the user to supply the value to represent "a" in the sequence "a+aa+aaa+aaaa";
1. To be clear, the user must output the sum of "9 + (9*9) + (9*9*9) + (9*9*9*9)" given a value of 9 NOT the sum of "9 + 99 + 999 + 9999";
1. Add input validation to ensure that the supplied value is an integer.

```py
def validate_input(n):
   return True if n.isdigit() else False

def get_input():
   n = input("Please input an integer value for 'a' to calculate the value of a+aa+aaa+aaaa. Only integers will be accepted:\t")
   while True:
       if (not validate_input(n)):
           n = input("You did not input a integer. Please do so:\t")
       else:
           break
   return int(n)

def calculate_value(n):
   a_counter = 0
   for elem in "a+aa+aaa+aaaa".split("+"):
      a_counter += n ** len(elem)
   print(a_counter)


if __name__ == "__main__":
   n = get_input()
   calculate_value(n)
```
