# DSC540

Week 1&2 assigment, Author: Kannur, Gyan

Task #1: create a list, iterate over the list and sort your results, generate random numbers, add to the list, and then print your results.

```
#import packages needed for all below exercises
import random
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from statistics import mean
import re


#create the list
max = 50
rand_nums = [x for x in range(0, max)]

#sort the list
num_sorted = rand_nums.sort(reverse=True)
print(num_sorted)
#get random number list, then sort and remove duplicate
random_list = [random.randint(-10, 0) for x in range(-10, 0)]

new_list = sorted(set(random_list))
combined_list = rand_nums + new_list
print(combined_list)
```

```
None
[49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33,
32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16,
15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -10, -9, -8, -4,
-3, -1]
```

Task #2: Create a line chart with Matplotlib and the following data file.

```
# the data file had been download into jupter notebook so read it
directly
pop_df = pd.read_excel('./datasets/world-population.xlsm')
pop_df
```

```
    Year  Population
0   1960  3028654024
1   1961  3068356747
2   1962  3121963107
3   1963  3187471383
4   1964  3253112403
5   1965  3320396924
```

```
6   1966  3390712300
7   1967  3460521851
8   1968  3531547287
9   1969  3606994959
10  1970  3682870688
11  1971  3761750672
12  1972  3839147707
13  1973  3915742695
14  1974  3992806090
15  1975  4068032705
16  1976  4141383058
17  1977  4214499013
18  1978  4288485981
19  1979  4363754326
20  1980  4439638086
21  1981  4516734312
22  1982  4595890494
23  1983  4675178812
24  1984  4753877875
25  1985  4834206631
26  1986  4918126890
27  1987  5004006066
28  1988  5090899475
29  1989  5178059174
30  1990  5266783430
31  1991  5351836347
32  1992  5433823608
33  1993  5516863641
34  1994  5598658151
35  1995  5681689325
36  1996  5762235749
37  1997  5842585301
38  1998  5921799957
39  1999  6001269553
40  2000  6078274622
41  2001  6155652495
42  2002  6232413711
43  2003  6309266583
44  2004  6385778679
45  2005  6462054420
46  2006  6538196688
47  2007  6614396907
48  2008  6692030277
49  2009  6775235741
```

```
#read the data, also create log value of population so it is more
read-able
year = pop_df['Year']
pop = pop_df['Population']
pop.head()
```

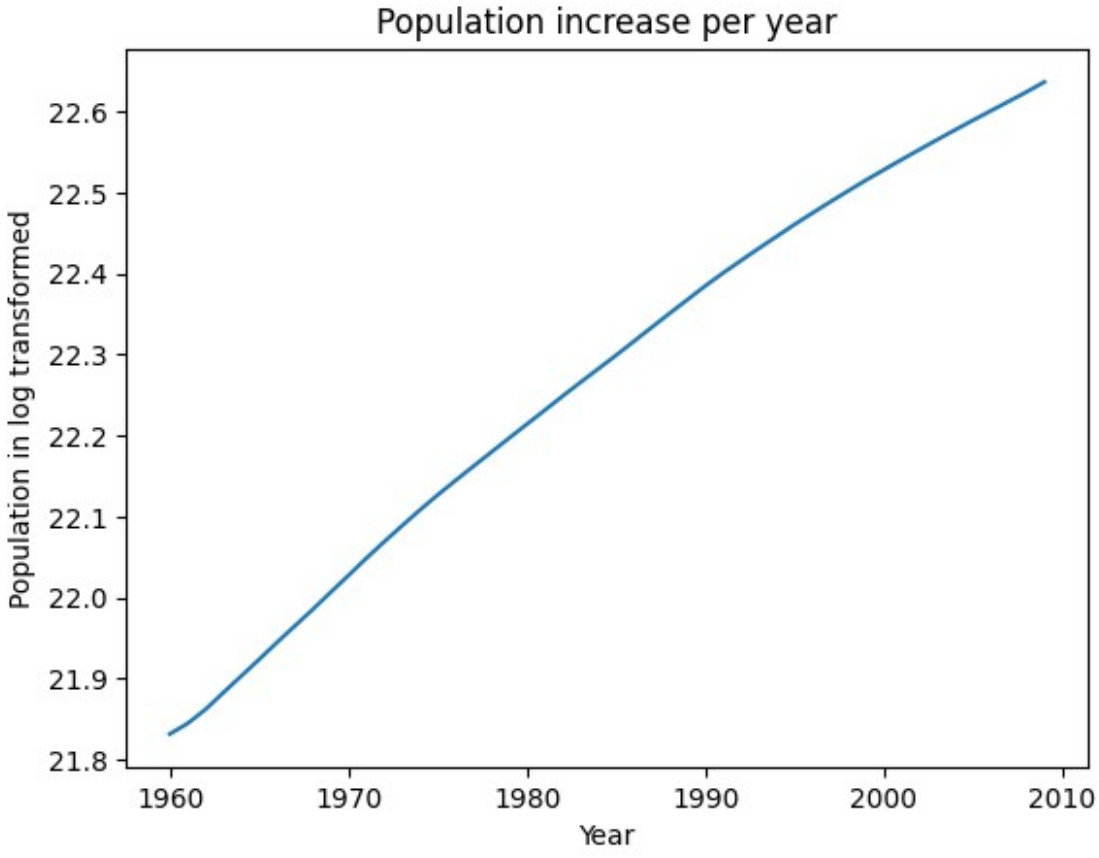
```
#Log transform
log_pop =np.log(pop)
log_pop
```

```
0     21.831384
1     21.844408
2     21.861728
3     21.882494
4     21.902878
5     21.923350
6     21.944306
7     21.964685
8     21.985002
9     22.006141
10    22.026958
11    22.048150
12    22.068516
13    22.088271
14    22.107760
15    22.126425
16    22.144296
17    22.161797
18    22.179200
19    22.196599
20    22.213839
21    22.231055
22    22.248428
23    22.265533
24    22.282227
25    22.298983
26    22.316194
27    22.333505
28    22.350720
29    22.367696
30    22.384686
31    22.400706
32    22.415909
33    22.431075
34    22.445793
35    22.460514
36    22.474591
37    22.488439
38    22.501906
39    22.515237
40    22.527987
41    22.540637
42    22.553030
43    22.565285
44    22.577339
45    22.589213
```

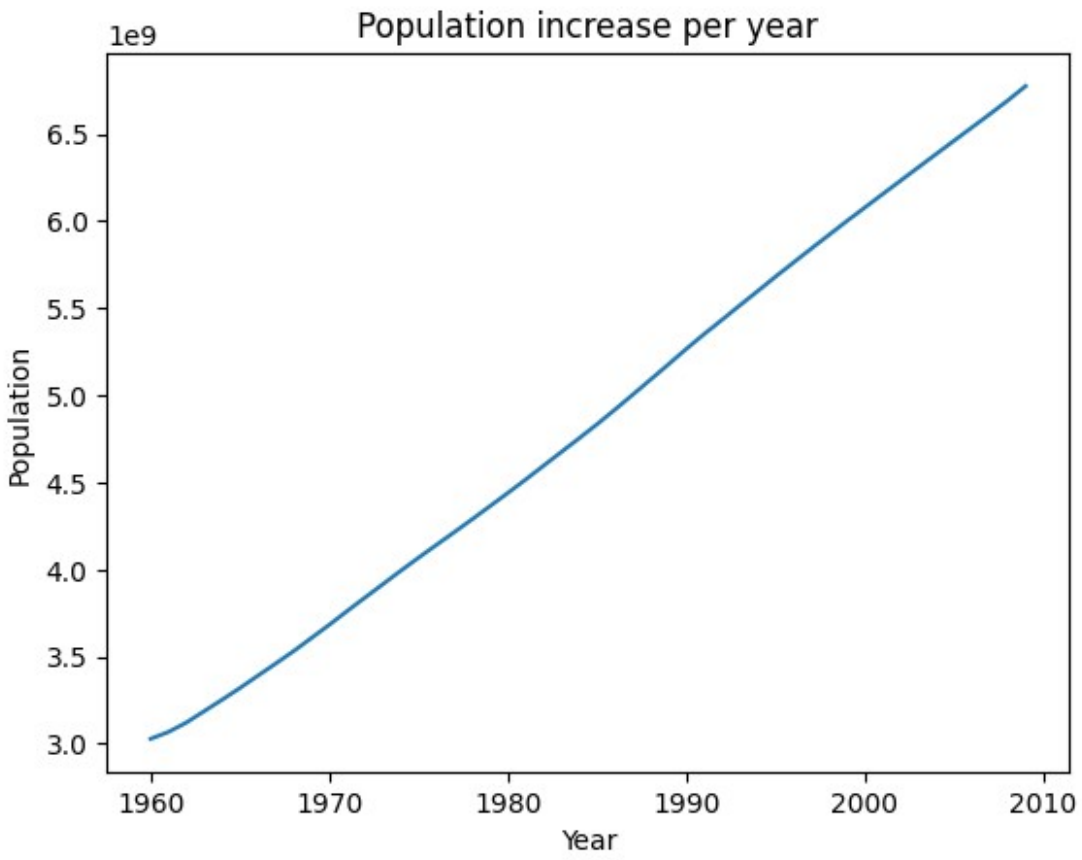
```
46    22.600927
47    22.612514
48    22.624183
49    22.636540
Name: Population, dtype: float64
```

```
#plot a line chart using log population value
plt.plot(year, log_pop)
plt.title('Population increase per year')
plt.xlabel('Year')
plt.ylabel('Population log transformed')
#plot a line chart using log populationplt.show()
```

```
Text(0, 0.5, 'Population in log transformed')
```




```
#plot a line chart using original population value
plt.plot(year, pop)
plt.title('Population increase per year')
plt.xlabel('Year')
plt.ylabel('Population')
plt.show()
```




Task 3.1: Data Wrangling with Python: Activity 1 handling list

```
#create a list of 100 random numbers
rand_list = [random.randint(0, 100) for x in range(0, 100)]

#Create a new list from this random list, with numbers that are
divisible by 3.
only_three_div_list = [a for a in rand_list if a % 3 == 0]

#Calculate the length of these two lists and store the difference in a
new variable.
len_difference = len(rand_list) - len(only_three_div_list)

#Using a loop, perform steps 2 and 3 and find the difference variable
three times
difference_list = []

for i in range(0, 3):
    list_new = [random.randint(0, max) for x in range(0, max)]
    only_three_div_list = [a for a in list_new if a % 3 == 0]
    difference = len(list_new) - len(only_three_div_list)
    difference_list.append(difference)
print("difference in length in 3 run are", difference_list)
print("mean of differences is", round(mean(difference_list),2))
```

```
difference in length in 3 run are [36, 32, 27]
mean of differences is 31.67
```

Task 3.2: Activity 2: Analyze a Multiline String and Generate the Unique Word Count

```
mutiline_text = """It is a truth universally acknowledged, that a
single man in possession of a good fortune, must be in want of a wife.

However little known the feelings or views of such a man may be on his
first entering a neighbourhood, this truth is so well fixed in the
minds of the surrounding families, that he is considered the rightful
property of some one or other of their daughters.

"My dear Mr. Bennet," said his lady to him one day, "have you heard
that Netherfield Park is let at last?"

Mr. Bennet replied that he had not.

"But it is," returned she; "for Mrs. Long has just been here, and she
told me all about it."

Mr. Bennet made no answer.

"Do you not want to know who has taken it?" cried his wife
impatiently.

"You want to tell me, and I have no objection to hearing it."

This was invitation enough.

"Why, my dear, you must know, Mrs. Long says that Netherfield is taken
by a young man of large fortune from the north of England; that he
came down on Monday in a chaise and four to see the place, and was so
much delighted with it, that he agreed with Mr. Morris immediately;
that he is to take possession before Michaelmas, and some of his
servants are to be in the house by the end of next week."

"What is his name?"

"Bingley."

"Is he married or single?"
"Oh! Single, my dear, to be sure! A single man of large fortune; four
or five thousand a year. What a fine thing for our girls!"

"How so? How can it affect them?"

"My dear Mr. Bennet," replied his wife, "how can you be so tiresome!
You must know that I am thinking of his marrying one of them."

"Is that his design in settling here?"
```

"Design! Nonsense, how can you talk so! But it is very likely that he may fall in love with one of them, and therefore you must visit him as soon as he comes."

"I see no occasion for that. You and the girls may go, or you may send them by themselves, which perhaps will be still better, for as you are as handsome as any of them, Mr. Bingley may like you the best of the party."

"My dear, you flatter me. I certainly have had my share of beauty, but I do not pretend to be anything extraordinary now. When a woman has five grown-up daughters, she ought to give over thinking of her own beauty."

"In such cases, a woman has not often much beauty to think of."

"But, my dear, you must indeed go and see Mr. Bingley when he comes into the neighbourhood."

"It is more than I engage for, I assure you."
"But consider your daughters. Only think what an establishment it would be for one of them. Sir William and Lady Lucas are determined to go, merely on that account, for in general, you know, they visit no newcomers. Indeed you must go, for it will be impossible for us to visit him if you do not."

"You are over-scrupulous, surely. I dare say Mr. Bingley will be very glad to see you; and I will send a few lines by you to assure him of my hearty consent to his marrying whichever he chooses of the girls; though I must throw in a good word for my little Lizzy."

"I desire you will do no such thing. Lizzy is not a bit better than the others; and I am sure she is not half so handsome as Jane, nor half so good-humoured as Lydia. But you are always giving her the preference."

"They have none of them much to recommend them," replied he; "they are all silly and ignorant like other girls; but Lizzy has something more of quickness than her sisters."

"Mr. Bennet, how can you abuse your own children in such a way? You take delight in vexing me. You have no compassion for my poor nerves."

"You mistake me, my dear. I have a high respect for your nerves. They are my old friends. I have heard you mention them with consideration these last twenty years at least."

"Ah, you do not know what I suffer."

```
"But I hope you will get over it, and live to see many young men of four thousand a year come into the neighbourhood."

"It will be no use to us, if twenty such should come, since you will not visit them."

"Depend upon it, my dear, that when there are twenty, I will visit them all."

Mr. Bennet was so odd a mixture of quick parts, sarcastic humour, reserve, and caprice, that the experience of three-and-twenty years"""

# 1. find type and length
print('type is', type(mutiline_text))

print('length is', len(mutiline_text))
```

```
type is <class 'str'>
length is 4137
```

```
# 2. Remove all new lines and symbols (inlcude ?,! etc) using the replace function
clean_text = re.sub(r'[?|$|.|!|"|,|;|:]',r'',mutiline_text)

# 3. Find all of the words in multiline_text using the split function.
list_of_words = clean_text.split()

# 4. create a list contains only unique words, seems
unique_words_as_list = list(set(list_of_words))

# 5. Count the number of times the unique word has appeared in the list using the key and value in dict.

#create dictionary first
unique_words_as_dict = dict.fromkeys(list_of_words)

#loop to get unique word count, if word is not exist, add, if exist, count +1
for word in list_of_words:
    if unique_words_as_dict[word] is None:
        unique_words_as_dict[word] = 1
    else:
        unique_words_as_dict[word] += 1
unique_words_as_dict
```

```
{'It': 3,
 'is': 12,
 'a': 19,
 'truth': 2,
 'universally': 1,
 'acknowledged': 1,
```

```
 'that': 15,
 'single': 3,
 'man': 4,
 'in': 11,
 'possession': 2,
 'of': 27,
 'good': 2,
 'fortune': 3,
 'must': 7,
 'be': 11,
 'want': 3,
 'wife': 3,
 'However': 1,
 'little': 2,
 'known': 1,
 'the': 17,
 'feelings': 1,
 'or': 5,
 'views': 1,
 'such': 5,
 'may': 5,
 'on': 3,
 'his': 9,
 'first': 1,
 'entering': 1,
 'neighbourhood': 3,
 'this': 1,
 'so': 8,
 'well': 1,
 'fixed': 1,
 'minds': 1,
 'surrounding': 1,
 'families': 1,
 'he': 11,
 'considered': 1,
 'rightful': 1,
 'property': 1,
 'some': 2,
 'one': 5,
 'other': 2,
 'their': 1,
 'daughters': 3,
 'My': 3,
 'dear': 8,
 'Mr': 10,
 'Bennet': 6,
 'said': 1,
 'lady': 1,
 'to': 19,
```

```
 'him': 4,
 'day': 1,
 'have': 7,
 'you': 24,
 'heard': 2,
 'Netherfield': 2,
 'Park': 1,
 'let': 1,
 'at': 2,
 'last': 2,
 'replied': 3,
 'had': 2,
 'not': 9,
 'But': 6,
 'it': 11,
 'returned': 1,
 'she': 4,
 'for': 12,
 'Mrs': 2,
 'Long': 2,
 'has': 5,
 'just': 1,
 'been': 1,
 'here': 2,
 'and': 14,
 'told': 1,
 'me': 5,
 'all': 3,
 'about': 1,
 'made': 1,
 'no': 7,
 'answer': 1,
 'Do': 1,
 'know': 5,
 'who': 1,
 'taken': 2,
 'cried': 1,
 'impatiently': 1,
 'You': 7,
 'tell': 1,
 'I': 17,
 'objection': 1,
 'hearing': 1,
 'This': 1,
 'was': 3,
 'invitation': 1,
 'enough': 1,
 'Why': 1,
 'my': 10,
```

```
 'says': 1,
 'by': 4,
 'young': 2,
 'large': 2,
 'from': 1,
 'north': 1,
 'England': 1,
 'came': 1,
 'down': 1,
 'Monday': 1,
 'chaise': 1,
 'four': 3,
 'see': 5,
 'place': 1,
 'much': 3,
 'delighted': 1,
 'with': 4,
 'agreed': 1,
 'Morris': 1,
 'immediately': 1,
 'take': 2,
 'before': 1,
 'Michaelmas': 1,
 'servants': 1,
 'are': 8,
 'house': 1,
 'end': 1,
 'next': 1,
 'week': 1,
 'What': 2,
 'name': 1,
 'Bingley': 4,
 'Is': 2,
 'married': 1,
 'Oh': 1,
 'Single': 1,
 'sure': 2,
 'A': 1,
 'five': 2,
 'thousand': 2,
 'year': 2,
 'fine': 1,
 'thing': 2,
 'our': 1,
 'girls': 4,
 'How': 2,
 'can': 4,
 'affect': 1,
 'them': 11,
```

```
 'how': 3,
 'tiresome': 1,
 'am': 2,
 'thinking': 2,
 'marrying': 2,
 'design': 1,
 'settling': 1,
 'Design': 1,
 'Nonsense': 1,
 'talk': 1,
 'very': 2,
 'likely': 1,
 'fall': 1,
 'love': 1,
 'therefore': 1,
 'visit': 5,
 'as': 7,
 'soon': 1,
 'comes': 2,
 'occasion': 1,
 'go': 4,
 'send': 2,
 'themselves': 1,
 'which': 1,
 'perhaps': 1,
 'will': 9,
 'still': 1,
 'better': 2,
 'handsome': 2,
 'any': 1,
 'like': 2,
 'best': 1,
 'party': 1,
 'flatter': 1,
 'certainly': 1,
 'share': 1,
 'beauty': 3,
 'but': 2,
 'do': 4,
 'pretend': 1,
 'anything': 1,
 'extraordinary': 1,
 'now': 1,
 'When': 1,
 'woman': 2,
 'grown-up': 1,
 'ought': 1,
 'give': 1,
 'over': 2,
```

```
 'her': 3,
 'own': 2,
 'In': 1,
 'cases': 1,
 'often': 1,
 'think': 2,
 'indeed': 1,
 'when': 2,
 'into': 2,
 'more': 2,
 'than': 3,
 'engage': 1,
 'assure': 2,
 'consider': 1,
 'your': 3,
 'Only': 1,
 'what': 2,
 'an': 1,
 'establishment': 1,
 'would': 1,
 'Sir': 1,
 'William': 1,
 'Lady': 1,
 'Lucas': 1,
 'determined': 1,
 'merely': 1,
 'account': 1,
 'general': 1,
 'they': 2,
 'newcomers': 1,
 'Indeed': 1,
 'impossible': 1,
 'us': 2,
 'if': 2,
 'over-scrupulous': 1,
 'surely': 1,
 'dare': 1,
 'say': 1,
 'glad': 1,
 'few': 1,
 'lines': 1,
 'hearty': 1,
 'consent': 1,
 'whichever': 1,
 'chooses': 1,
 'though': 1,
 'throw': 1,
 'word': 1,
 'Lizzy': 3,
```

```
 'desire': 1,
 'bit': 1,
 'others': 1,
 'half': 2,
 'Jane': 1,
 'nor': 1,
 'good-humoured': 1,
 'Lydia': 1,
 'always': 1,
 'giving': 1,
 'preference': 1,
 'They': 2,
 'none': 1,
 'recommend': 1,
 'silly': 1,
 'ignorant': 1,
 'something': 1,
 'quickness': 1,
 'sisters': 1,
 'abuse': 1,
 'children': 1,
 'way': 1,
 'delight': 1,
 'vexing': 1,
 'compassion': 1,
 'poor': 1,
 'nerves': 2,
 'mistake': 1,
 'high': 1,
 'respect': 1,
 'old': 1,
 'friends': 1,
 'mention': 1,
 'consideration': 1,
 'these': 1,
 'twenty': 3,
 'years': 2,
 'least': 1,
 'Ah': 1,
 'suffer': 1,
 'hope': 1,
 'get': 1,
 'live': 1,
 'many': 1,
 'men': 1,
 'come': 2,
 'use': 1,
 'should': 1,
 'since': 1,
```

```
 'Depend': 1,
 'upon': 1,
 'there': 1,
 'odd': 1,
 'mixture': 1,
 'quick': 1,
 'parts': 1,
 'sarcastic': 1,
 'humour': 1,
 'reserve': 1,
 'caprice': 1,
 'experience': 1,
 'three-and-twenty': 1}
```

```
# 4. Find the top 25 words from the unique words that you have found
using the slice function.
# using word count as sorted key, sort from largest to smallest,
result is a dictionary of tuple pair
top_25words = sorted(unique_words_as_dict.items(), key=lambda
key_val_tuple: key_val_tuple[1], reverse=True)
top_25words[:25]
```

```
[('of', 27),
 ('you', 24),
 ('a', 19),
 ('to', 19),
 ('the', 17),
 ('I', 17),
 ('that', 15),
 ('and', 14),
 ('is', 12),
 ('for', 12),
 ('in', 11),
 ('be', 11),
 ('he', 11),
 ('it', 11),
 ('them', 11),
 ('Mr', 10),
 ('my', 10),
 ('his', 9),
 ('not', 9),
 ('will', 9),
 ('so', 8),
 ('dear', 8),
 ('are', 8),
 ('must', 7),
 ('have', 7)]
```

Task 3.3 Activity 3: Permutation, Iterator, Lambda, List

```
#import package
from itertools import permutations, dropwhile

# 1. look up definition of permutations and dropwhile
#permutations?
#dropwhile?
#permutations: Return successive r-length permutations of elements in
the iterable.
#dropwhile: returns an iterator only after the func . in argument
returns false for the first time.

# 2. Write an expression to generate all the possible three-digit
numbers using 0, 1, and 2.
# 3.  Use assert and isinstance to make sure that the elements are of
the tuple type.
perm = permutations([0, 1, 2])
for i in list(perm):
    print (i)
    assert isinstance(i, tuple)
```

```
(0, 1, 2)
(0, 2, 1)
(1, 0, 2)
(1, 2, 0)
(2, 0, 1)
(2, 1, 0)
```

```
# 4. Write the loop again using dropwhile with a lambda expression to
drop any leading zeros from the tuples.
for t in permutations(range(3)):
    print(list(dropwhile(lambda x: x <= 0, t)))
```

```
[1, 2]
[2, 1]
[1, 0, 2]
[1, 2, 0]
[2, 0, 1]
[2, 1, 0]
```

```
# 5. Check the actual type that dropwhile returns
# permutations created tuple, the result of drop while is still a
tuple. then 'list conversion' used outside the dropwhile, the result
is a list

#6. Combine the preceding code into one block, and this time write a
separate function

"""
this function 'number_conversion' will take each group of numbers in
```

```
the returned list, the len is the power order of 10, then
multiply each number with the power, then add together to get sum
so [2,1] will = 2x10 + 1
"""

import math

def number_conversion(number_stack):
    final_number = 0
    for i in range(0, len(number_stack)):
        final_number += (number_stack.pop() * (math.pow(10, i)))
    return final_number

for t in permutations(range(3)):
    number_stack = list(dropwhile(lambda x: x <= 0, t))
    print(number_conversion(number_stack))
```

```
12.0
21.0
102.0
120.0
201.0
210.0
```

Task 3.4 Activity 4: Design Your Own CSV Parser

```
# csvfile had been download and uploaded into juypter notebook
#import package

from itertools import zip_longest

''' use zip_longest function to take each element in header list and
line list (each row) alternatively, so turn them into tuple pairs
    then read them as dictionary key and value

'''
def return_dict_from_csv_line(header, line):
    zipped_line = zip_longest(header, line, fillvalue=None)
    # read zipped_line for key and value, to generate the final dict
    ret_dict = {kv[0]: kv[1] for kv in zipped_line}
    return ret_dict

''' open file first, read each line,
    take first line as header by removing line end, then using ',' to
get each word into a list named 'header';
    for all following lines, remove new lines and use ',' to turn the
each word in a list named 'line';
    use enumerate function to get a counter of lines.
    to save space, only read 3 line of records.

'''
```

```
with open("./datasets/sales_record.csv", "r") as fd:
    first_line = fd.readline()
    header = first_line.replace("\n", "").split(",")
    for i, line in enumerate(fd):
        # loop over the first 3 lines only, i starts with 0 , will run
3 times.
        if i > 2:
            break
        line = line.replace("\n", "").split(",")
        d = return_dict_from_csv_line(header, line)
        print(d)
```

```
{'Region': 'Central America and the Caribbean', 'Country': 'Antigua
and Barbuda ', 'Item Type': 'Baby Food', 'Sales Channel': 'Online',
'Order Priority': 'M', 'Order Date': '12/20/2013', 'Order ID':
'957081544', 'Ship Date': '1/11/2014', 'Units Sold': '552', 'Unit
Price': '255.28', 'Unit Cost': '159.42', 'Total Revenue': '140914.56',
'Total Cost': '87999.84', 'Total Profit': '52914.72'}
{'Region': 'Central America and the Caribbean', 'Country': 'Panama',
'Item Type': 'Snacks', 'Sales Channel': 'Offline', 'Order Priority':
'C', 'Order Date': '7/5/2010', 'Order ID': '301644504', 'Ship Date':
'7/26/2010', 'Units Sold': '2167', 'Unit Price': '152.58', 'Unit
Cost': '97.44', 'Total Revenue': '330640.86', 'Total Cost':
'211152.48', 'Total Profit': '119488.38'}
{'Region': 'Europe', 'Country': 'Czech Republic', 'Item Type':
'Beverages', 'Sales Channel': 'Offline', 'Order Priority': 'C', 'Order
Date': '9/12/2011', 'Order ID': '478051030', 'Ship Date': '9/29/2011',
'Units Sold': '4778', 'Unit Price': '47.45', 'Unit Cost': '31.79',
'Total Revenue': '226716.10', 'Total Cost': '151892.62', 'Total
Profit': '74823.48'}
```