Homework number 1

August 6, 2019

1 Exercise

Generate odd numbers. Write a program that generates all odd numbers from 1 to n. Set n in the beginning of the program and use a while loop to compute the numbers. (Make sure that if n is an even number, the largest generated odd number is n-1.).

```
In [33]: print("Please et an integer number:")
         x = input()
         x=int(x)
         print('The odd numbers between 1 and', x,'are:')
         i = 1
         while i <= x:
           print(i)
           i += 2
```

```
Please et an integer number:
7
The odd numbers between 1 and 7 are:
1
3
5
7
```

2 Exercise

Store odd numbers in a list. Modify the program from the previous exercise to store the generated odd numbers in a list. Start with an empty list and use a while loop where you in each pass of the loop append a new element to the list. Finally, print the list elements to the screen.

```
In [34]: print("Please set an integer number:")
         x = input()
         x=int(x)
         print('The odd numbers between 1 and', x,'are:')
         i=1
         numbers =[]
         while i <= x:
                numbers.append(i)
                i += 2
            print(numbers)
```

```
Please set an integer number:
7
The odd numbers between 1 and 7 are:
[1, 3, 5, 7]
```

3 Exercise

Generate odd numbers by a list comprehension. Solve Exercise 2 using a list comprehension (with for and range).

```
In [35]: print("Please set an integer number:")
         x = input()
         x=int(x)
         print('The odd numbers between 1 and', x,'are:')
         numbers =[]
         a = x + 1
         for b in range(1, a, 2):
             numbers.append(b)
         print(numbers)
```

```
Please set an integer number:
7
The odd numbers between 1 and 7 are:
[1, 3, 5, 7]
```

4 Exercise

Make a table of values from a formula. Write a program that prints a nicely formatted table of t and y(t) values, where
   Use n uniformly spaced t values throughout the interval [0, 2v0/g]. Set v0 =1 and n = 11.

```
In [36]: from numpy import arange
         Vo = 1
         g = 9.81
         A = 2 * Vo / 9.81
         B = A / 11
         print('{:21}{:21}'.format('          t          ','          y          '))
         for t in arange (B, A+B, B):
             y = Vo * t - 0.5 * g * t * t
             print('{:21}{:21}'.format(t,y))
```

```
          t                     y
  0.01853396348809193 0.016849057716447208
  0.03706792697618386 0.030328303889604974
 0.055601890464275786   0.0404377385194733
  0.07413585395236771  0.04717736160605218
  0.09266981744045964 0.050547173149341616
  0.11120378092855157 0.050547173149341616
  0.12973774441664349  0.04717736160605218
  0.14827170790473543   0.0404377385194733
  0.16680567139282737 0.030328303889604946
   0.1853396348809193  0.01684905771644718
   0.2038735983690112                  0.0
```

5 Exercise

Store values from a formula in lists. Modify the program from Exercise 4 so that the t and y values are stored in two lists t and y. Thereafter, transverse the lists with a for loop and write out a nicely formatted table of t and y values (using either a zip or range construction). Set v0 =10 and n = 81.

```
In [37]: from numpy import arange
         Vo = 10
         g = 9.81
         A = 2 * Vo / 9.81
         B = A / 81
         tlist = []
         ylist = []
         i=1
         print('{:21}{:21}'.format('          t           ','           y           '))
         for t in arange (B, A+B, B):
             y = Vo * t - 0.5 * g * t * t
             print('{:21}{:21}'.format(t,y))
             tlist.append(t)
             ylist.append(y)
```

```
          t                     y
 0.025169580045556937  0.24858844489438953
 0.050339160091113874   0.4909621786664193
   0.0755087401366708   0.7271212013160893
  0.10067832018222775   0.9570655128433997
   0.1258479002277847   1.1807951132483503
  0.15101748027334164    1.398310002530941
  0.17618706031889855    1.609610180691172
   0.2013566403644555   1.8146956477290435
  0.22652622041001244    2.013566403644555
   0.2516958004555694    2.206222448437707
   0.2768653805011263    2.392663782108499
   0.3020349605466832    2.572890404656931
```

0.32720454059224013	2.7469023160830037
0.3523741206377971	2.9146995163867166
0.377543700683354	3.0762820055680695
0.402713280728911	3.2316497836270637
0.4278828607744679	3.3808028505636973
0.4530524408200248	3.523741206377971
0.4782220208655818	3.660464851069885
0.5033916009111388	3.7909737846394402
0.5285611809566957	3.9152680070866346
0.5537307610022526	4.033347518411469
0.5789003410478095	4.145212318613945
0.6040699210933664	4.25086240769406
0.6292395011389234	4.350297785651816
0.6544090811844804	4.443518452487212
0.6795786612300373	4.5305244082002485
0.7047482412755942	4.611315652790925
0.7299178213211511	4.685892186259242
0.755087401366708	4.754254008605199
0.7802569814122651	4.816401119828797
0.805426561457822	4.872333519930034
0.8305961415033789	4.922051208908912
0.8557657215489358	4.965554186765431
0.8809353015944927	5.002842453499588
0.9061048816400498	5.033916009111387
0.9312744616856067	5.058774853600827
0.9564440417311636	5.077418986967905
0.9816136217767205	5.0898484092126255
1.0067832018222775	5.0960631203349855
1.0319527818678345	5.0960631203349855
1.0571223619133914	5.0898484092126255
1.0822919419589483	5.077418986967905
1.1074615220045052	5.058774853600827
1.1326311020500621	5.033916009111389
1.157800682095619	5.002842453499589
1.182970262141176	4.96555418676543
1.2081398421867329	4.922051208908913
1.2333094222322898	4.872333519930035
1.258479002277847	4.816401119828797
1.2836485823234038	4.7542540086052
1.3088181623689608	4.685892186259242
1.3339877424145177	4.6113156527909265
1.3591573224600746	4.5305244082002485
1.3843269025056315	4.443518452487213
1.4094964825511884	4.350297785651817
1.4346660625967453	4.250862407694061
1.4598356426423023	4.145212318613945
1.4850052226878592	4.033347518411471
1.5101748027334163	3.9152680070866364

```
1.5353443827789732   3.7909737846394407
1.5605139628245301    3.660464851069886
 1.585683542870087    3.523741206377972
 1.610853122915644   3.3808028505636987
 1.636022702961201    3.231649783627063
1.6611922830067578   3.0762820055680713
1.6863618630523147    2.914699516386719
1.7115314430978716   2.7469023160830055
1.7367010231434286    2.572890404656933
1.7618706031889855   2.3926637821084995
1.7870401832345426    2.206222448437707
1.8122097632800995   2.0135664036445533
1.8373793433256564    1.814695647729046
1.8625489233712134    1.609610180691174
1.8877185034167703   1.3983100025309412
1.9128880834623272    1.180795113248351
 1.938057663507884   0.9570655128434034
 1.963227243553441   0.7271212013160913
 1.988396823598998   0.4909621786664218
 2.013566403644555  0.24858844489439136
 2.038735983690112                  0.0
```

6 Exercise

Work with a list. Set a variable primes to a list containing the numbers 2, 3, 5, 7,11, and 13. Write out each list element in a for loop. Assign 17 to a variable p and add p to the end of the list. Print out the whole new list.

```
In [38]: from random import choice
         list = [2,3,5,7,11,13]
         Nlist = []
         p = 17
         for i in range(0,6,1):
             Nlist.append(list[i])
         Nlist.append(p)
         print(list)
         print(Nlist)
```

```
[2, 3, 5, 7, 11, 13]
[2, 3, 5, 7, 11, 13, 17]
```

7 Exercise

Simulate operations on lists by hand. You are given the following program:

```
In [39]: a = [1,3,5,7,11]         #Creates the 'a' list
         b = [13,17]              #Creates the 'b' list
         c = a + b                #Adds to the 'a' list the characters in
                                  #the 'b' list in a new 'c' list
         print(c)                 #Shows the 'c' list
         b[0] = -1                #Replaces the first character in 'b' list with -1
         d = [e+1 for e in a]     #Saves in a 'd' list the characters in the 'a'
                                  #list with the same numbers but added eachone by one
         print(d)                 #Shows the 'd' list
         d.append(b[0] + 1)       #Adds to the 'd' list the first character
                                  #in 'b' list added by one
         d.append(b[-1] + 1)      #Adds to the 'd' list the last character in
                                  #'b' list added by one
         print(d[-2:])            #Shows the 'd' list starting from the
                                  #penultimate value onwords
```

```
[1, 3, 5, 7, 11, 13, 17]
[2, 4, 6, 8, 12]
[0, 18]
```