W18 Reading Unit 4

June 19, 2016

1 1. For Loops

For printing purpose, we can do print(x, end = “”) to avoid change in line

1.1 1.1 an example of print(x,end=“”)

```
In [26]: x = "name"

          for a in x:
              print(a)
          print("--------")

          for b in x:
              print(b , end = "")
```

```
n
a
m
e
--------
name
```

1.2 1.2 While and For countdown

```
In [1]: countdown = 5
        while countdown > 0:
            print(countdown)
            countdown -=1
        print("Blast off!")
```

```
5
4
3
2
1
Blast off!
```

```
In [4]: countdown = 5
        for x in range(countdown,0,-1):
        # range(a,b,c) : a range starting from countdown uptill b but not
        # including 0 with step size -1
            print(x)
        print("Blast off!")
```

```
5
4
3
2
1
Blast off!
```

1.3 1.3 While and For nested

```
In [10]: row = int(input("enter an integer: "))
         while row >=0:
             # inner loop
             j = 0
             while j <= row:
                 print(j,end = " ")
                 j += 1
             print("")
             row -=1
```

```
enter an integer: 5
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

```
In [20]: row = int(input("enter an integer: "))
         for x in range(row,-1,-1):
             for y in range(0,x+1,1):
                 print(y,end = " ")
             print(" ")
```

```
enter an integer: 5
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
```

```
In [30]: x = input("Enter your name: ")
         for char in x:
             if char.lower() not in "aeiou":
                 print(char, end = "")
```

```
Enter your name: Paul
Pl
```

```
In [33]: x = ["Paul", "Bill", "Kay"]
         for name in x:
             print(name + ", as himself")
```

```
Paul, as himself
Bill, as himself
Kay, as himself
```

2 2. Fancy Loops Exits

2.1 2.1 Bool Flag

we want to end the loop earlier!

```
In [39]: x = int(input("Enter a number: "))
         prime = True
         for i in range(2,x):
             #check if i divides x.
             if x % i == 0:
                 # now I know x is not prime
                 prime = False
         if prime == True:
             print(x, "is prime.")
         else:
             print(x,"is not a prime.")
```

```
Enter a number: 2000
2000 is not a prime.
```

```
In [45]: x = int(input("Enter a number: "))
         prime = True

         i = 2
         while i < x and prime == True:
         # since prime is a bool and it is initially true, we can simply do
         # while i < x and prime :
             #check if i divides x.
             if x % i == 0:
                 # now I know x is not prime
                 prime = False
             i += 1
         print("kept looping until i is", i)
         if prime == True:
             print(x, "is prime.")
         else:
             print(x,"is not a prime.")
```

```
Enter a number: 2000
kept looping until i is 3
2000 is not a prime.
```

2.2 2.2 Break

```
In [3]: x = int(input("Enter a number: "))
        prime = True
        for i in range(2,x):
            #check if i divides x.
            print("Now checking possible divisor", i)
            if x % i == 0:
                # now I know x is not prime
                prime = False
                break
        if prime == True:
            print(x, "is prime.")
        else:
            print(x,"is not a prime.")
```

```
Enter a number: 13
Now checking possible divisor 2
Now checking possible divisor 3
Now checking possible divisor 4
Now checking possible divisor 5
Now checking possible divisor 6
Now checking possible divisor 7
Now checking possible divisor 8
Now checking possible divisor 9
Now checking possible divisor 10
Now checking possible divisor 11
Now checking possible divisor 12
13 is prime.
```

2.3 2.3 else in For

else only executed when it ends smoothly,r.g. without going to a break

```
In [12]: x = int(input("Enter a number: "))

         for i in range(2,x):
             # check if i divides x.
             print("Now checking possible divisor", i)
             if x % i == 0:
                 # now I know x is not prime
                 print(x, "is not a prime.")
                 break
         else:
             print(x, "is prime.")
```

```
Enter a number: 15
Now checking possible divisor 2
Now checking possible divisor 3
15 is not a prime.
```

2.4 2.4 perfome a basic check before the loop

e.g. are the vowels in a word in alphabetical order?

```
In [18]: word = input("Enter a word: ").lower()
         last = "a"

         for letter in word:
             if letter in "aeiou":
                 print(letter)
                 # check to see if order looks good

                 if letter < last:
                     print("The vowels in", word, "are out of order.")
                     break
                 last = letter
         else:
             print("The vowels in", word, "are in order.")
```

```
Enter a word: Paul
a
u
The vowels in paul are in order.
```

2.5 2.5 Continue Statement

e.g. parallal checking conditions. what continue does is to skip the rest of this one loop

```
In [20]: word = input("Enter a word: ").lower()
         last = "a"

         for letter in word:
             if letter not in "aeiou":
                 continue
         # what continue does is to skip the rest of this one loop and go back to
         # the for statement!

             print(letter)

             # check to see if order looks good

             if letter < last:
                 print("The vowels in", word, "are out of order.")
                 break
             last = letter
         else:
             print("The vowels in", word, "are in order.")
```

```
Enter a word: llks
The vowels in llks are in order.
```

3 3. Algorithms

algorithm: 1. abstract set of steps a progra will execute 2. plan for solving a problem
problem: how do you find the square root of x with x>0 want to be within esillon of the real square root.
x ** 0.5

3.1 3.1 Exhaustive search: Brute Force

```
In [26]: x = float(input("Enter a number: "))

         num_guesses = 0
         epsilon = 0.00001
         ans = 0.0

         while ans ** 2 <= x:
             ans += epsilon
             num_guesses += 1
         print("number of guesses", num_guesses)
         print(ans, "is close to the square root of", x)
```

```
Enter a number: 12345
number of guesses 11110806
111.1080600240677 is close to the square root of 12345.0
```

3.2 3.2 Bisection Search

```
In [2]: x = float(input("Enter a number: "))

        epsilon = 0.00001
        num_guesses = 0
        low = 0
        high = x
        ans = (high + low)/2.0

        while high-low >= 2 * epsilon:
            print("low = ", low, "high= ", high)
            num_guesses += 1
            if ans**2 < x:
                low = ans
            else:
                high= ans
            ans = (high+low)/2.0
        print(ans,"is close to the square root of", x)
        print("Number of guesses:",num_guesses)
```

```
Enter a number: 12345
low =  0 high=  12345.0
low =  0 high=  6172.5
low =  0 high=  3086.25
low =  0 high=  1543.125
low =  0 high=  771.5625
low =  0 high=  385.78125
low =  0 high=  192.890625
low =  96.4453125 high=  192.890625
low =  96.4453125 high=  144.66796875
low =  96.4453125 high=  120.556640625
low =  108.5009765625 high=  120.556640625
low =  108.5009765625 high=  114.52880859375
low =  108.5009765625 high=  111.514892578125
low =  110.0079345703125 high=  111.514892578125
low =  110.76141357421875 high=  111.514892578125
low =  110.76141357421875 high=  111.13815307617188
low =  110.94978332519531 high=  111.13815307617188
low =  111.0439682006836 high=  111.13815307617188
low =  111.09106063842773 high=  111.13815307617188
low =  111.09106063842773 high=  111.1146068572998
low =  111.10283374786377 high=  111.1146068572998
low =  111.10283374786377 high=  111.10872030258179
low =  111.10577702522278 high=  111.10872030258179
low =  111.10724866390228 high=  111.10872030258179
low =  111.10798448324203 high=  111.10872030258179
low =  111.10798448324203 high=  111.10835239291191
low =  111.10798448324203 high=  111.10816843807697
low =  111.10798448324203 high=  111.1080764606595
low =  111.10803047195077 high=  111.1080764606595
low =  111.10805346630514 high=  111.1080764606595
111.10805921489373 is close to the square root of 12345.0
Number of guesses: 30
```

3.3 3.3 Heron's Method

find ans close to the square root of x
ans consider x/ans and ans* (x/ans) = x
next guess: ans = (x/ans + ans) / 2
A special case of Newton's Method. let f(ans) = ans ** 2 -x find ans such that f(ans) = 0

```
In [4]: x = float(input("Enter a number: "))

        eqsilon = 0.00001
        num_guesses = 0
        ans = 1

        while abs(x/ans - ans) > epsilon:
            ans = (x/ans + ans) / 2
            num_guesses += 1
        print(ans, "is close to the square root of,", x)
        print("Number of guesses: ", num_guesses)
```

```
Enter a number: 12345
111.10805770848404 is close to the square root of, 12345.0
Number of guesses:  10
```

4 4. Array

Array is not built in basic Python!
it is available in numpy, a very important package
in order to use a package, we need to import it
e.g. import numpy
or sometimes import numpy as np
Arrays are used to optimized for computational performance

```
In [2]: import numpy as np

In [4]: # create an array!
        np.arange(0,20)

Out[4]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
               17, 18, 19])

In [7]: x = np.arange(0,20)
        x

Out[7]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
               17, 18, 19])

In [8]: x[2] = 12
        x

Out[8]: array([ 0,  1, 12,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
               17, 18, 19])

In [11]: # check for datatype
         # every elements in an array has to be in the same type
         x.dtype

Out[11]: dtype('int32')
```

```
In [12]: # e.g if I do this will give an error
         x[10] = "shoe"


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-12-e6d50a5b8a15> in <module>()
          1 # e.g if I do this will give an error
    ----> 2 x[10] = "shoe"


        ValueError: invalid literal for int() with base 10: 'shoe'
```

```
In [13]: # the length of an array is fixed !!!!!!
         # e.g. the following two will give errors
         x.append(3)
         x.pop


        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        <ipython-input-13-e67d644ad88d> in <module>()
          1 # the length of an array is fixed !!!!!!
          2 # e.g. the following two will give errors
    ----> 3 x.append(3)
          4 x.pop


        AttributeError: 'numpy.ndarray' object has no attribute 'append'
```

```
In [16]: # asking for a 5*5 random array
         # notice the numbers are from 0 to 1, because they are getting from a
         # uniform distribution
         np.random.rand(5,5)

Out[16]: array([[ 0.43412768,  0.70650482,  0.57834829,  0.30760445,  0.66411115],
                [ 0.68718037,  0.64103205,  0.37015391,  0.53455498,  0.69730488],
                [ 0.24446339,  0.51349599,  0.5594937 ,  0.31504148,  0.57464194],
                [ 0.97181812,  0.31563402,  0.60238264,  0.6208236 ,  0.5138505 ],
                [ 0.94491418,  0.12789558,  0.13728023,  0.87259091,  0.63355134]])
```

```
In [17]: #asking for help
         np.random?
```

```
In [29]: ar = np.random.randn(20)
         ar

Out[29]: array([ 1.59924634, -1.14521362, -0.78834045,  0.58941176,  0.80687964,
               -0.0490016 ,  0.76980544,  1.34178542,  0.41562246,  0.26707351,
               -0.91954996,  0.68791742,  0.96076847,  1.12851396, -0.36312592,
               -0.7124818 ,  0.04828918,  0.79401736, -1.75099179, -0.70485244])
```

```
In [31]: m = ar.mean()
         s = ar.std()
         print("ar.mean is ",m,"\nar.std is",s )

ar.mean is  0.148788668611
ar.std is 0.895439586514
```

```
In [33]: # I can use reshape to give a new set of dimension
         ar = ar.reshape(5,4)
         ar

Out[33]: array([[ 1.59924634, -1.14521362, -0.78834045,  0.58941176],
                [ 0.80687964, -0.0490016 ,  0.76980544,  1.34178542],
                [ 0.41562246,  0.26707351, -0.91954996,  0.68791742],
                [ 0.96076847,  1.12851396, -0.36312592, -0.7124818 ],
                [ 0.04828918,  0.79401736, -1.75099179, -0.70485244]])
```

```
In [35]: # if I want the mean of each collom I can do
         mc = ar.mean(axis = 0)
         # if I want the mean of each row I can do
         mr = ar.mean(axis = 1)
         print("mean of each collom is ",mc,"\nmean of each row is ",mr )

mean of each collom is  [ 0.76616122  0.19907792 -0.61044054  0.24035607]
mean of each row is  [ 0.06377601  0.71736723  0.11276586  0.25341868 -0.40338442]
```

5 5. Comprehensions

```
In [37]: squares = []
         for i in range(1,11):
             squares.append(i**2)
         print(squares)

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [38]: # lets take a look at the samething but using Comprehensions
         [i**2 for i in range(1,11)]

Out[38]: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
In [41]: # nested loop using comprehensions
         [(row,column)for row in range(4) for column in range(4)]

Out[41]: [(0, 0),
          (0, 1),
          (0, 2),
          (0, 3),
          (1, 0),
          (1, 1),
          (1, 2),
          (1, 3),
          (2, 0),
          (2, 1),
          (2, 2),
          (2, 3),
          (3, 0),
            (3, 1),
            (3, 2),
            (3, 3)]
```

```
In [43]: import string
         letters = [a for a in string.ascii_lowercase]
         print(letters)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u
```

```
In [45]: # find all the letters in a word and put into alphabetical order
         import string
         word = input("Enter a word: ")
         letters = [a for a in string.ascii_lowercase if a in word]
         print(letters)
```

```
Enter a word: William
['a', 'i', 'l', 'm']
```

```
In [46]: # dictionary comprihension
         text = "Here is some exampe text."
         frequencies = {letter : text.count(letter) for letter in text }
         print(frequencies)
```

```
{'a': 1, 'x': 2, 'i': 1, ' ': 4, 'H': 1, 'o': 1, 'e': 6, 'r': 1, 'p': 1, 'm': 2, 's': 2, 't': 2, '.': 1
```