Lecture 03 & 04 课后习题

Python 101: Microsoft Exam 98-381

James W. Jiang 姜伟生 | Summer 2018, SavvyPro, Toronto, Canada

Question 1. What are the results from the following cells of programs?

```
if 1 == 1 and 2 == 2:

    print("True!")
```

```
if 1 == 2 and 2 == 2:

    print("True")
```

```
if 1 == 2 and 2 == 2:

    print("Second level")

print("Done")
```

```
if 1 == 2 or 2 == 3:

    print("True!")
```

```
if not False:

    print("True!")
```

```
if 30 == 30:

    print("30 equals 30!")

    if True:

        print("True is True!")
```

```
if 30 == 30:

    print("30 equals 30!")

    if False:

        print("True is True!")

print("Done")
```

Question 2. What are the results from the following cells of programs?

```
try:

    print(spam)

except NameError:

    print("spam isn't defined!")

except:

    print("An unknown error has occurred!")
```

```
try:

    spam = 7

    print("I have %d cans of spam!" % spam)

except:

    print("Some error has occurred!")

else:

    print("Everything went smoothly!")
```

```
try:

    print("pop")

except:

    print("An error has occurred!")

else:
```

```
    print("Everything went smoothly!")
finally:
    print("Finishing up now...")
```

```
try :
    2 / 0
except ZeroDivisionError :
    print ('Division by 0 detected.')
except :
    exit (0)
```

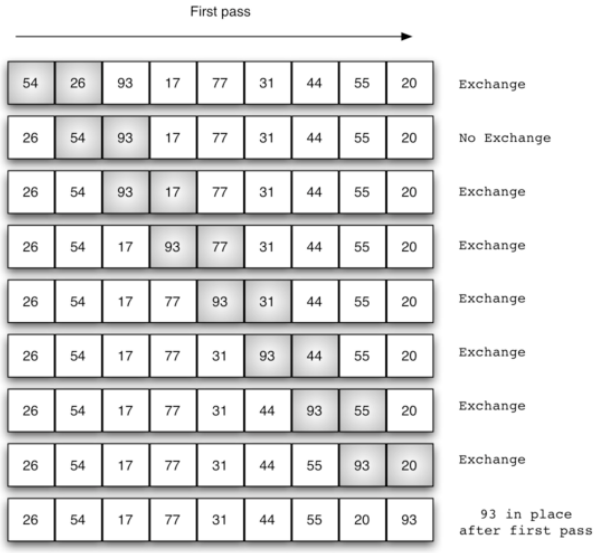
```
try:
    file = open("hello.txt", "r")
    print(file.read())
except IOError:
    print("An I/O error has occurred!")
except:
    print("An unknown error has occurred!")
finally:
    file.close()
```

Question 3. Find Prime Numbers in a Range: A positive integer greater than 1 which has no other factors except 1 and the number itself is called a prime number.
2, 3, 5, 7 etc. are prime numbers as they do not have any other factors. But 6 is not prime (it is composite) since, 2 x 3 = 6.

```
# Python program to display all the prime numbers within an interval


# change the values of lower and upper for a different result
```

```
lower = 900

upper = 1000


# uncomment the following lines to take input from the user

#lower = int(input("Enter lower range: "))

#upper = int(input("Enter upper range: "))


print("Prime numbers between",lower,"and",upper,"are:")


for num in range(lower,upper + 1):

   # prime numbers are greater than 1

   if num > 1:

       for i in range(2,num):

           if (num % i) == 0:

               break

       else:

           print(num)
```

Question 4. Calculating the Sum of a List of Numbers. We will begin our investigation with a simple problem that you already know how to solve without using recursion. Suppose that you want to calculate the sum of a list of numbers such as: [1,3,5,7,9]. An iterative function that computes the sum is shown in the first block. The second block uses a series of recursive calls. Compare the two algorithms.




```
def listsum(numList):
    theSum = 0
    for i in numList:
        theSum = theSum + i
    return theSum

print(listsum([1,3,5,7,9]))
```

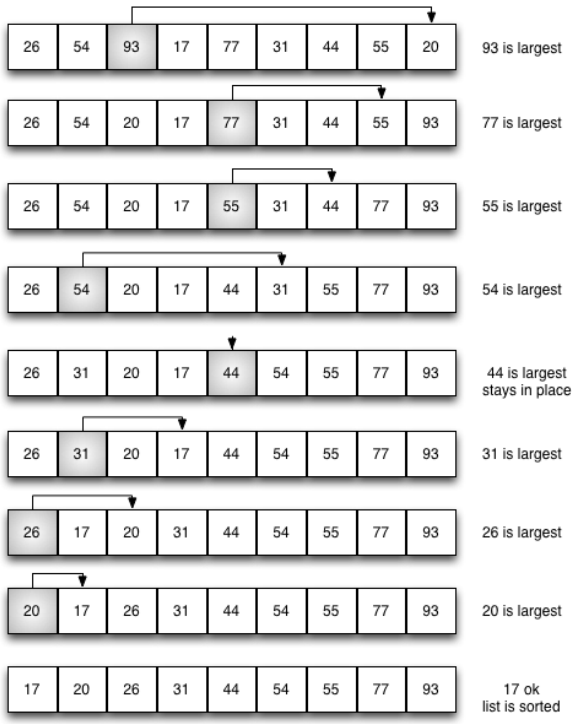
```
def listsum(numList):
   if len(numList) == 1:
        return numList[0]
   else:
        return numList[0] + listsum(numList[1:])

print(listsum([1,3,5,7,9]))
```

Question 5. The bubble sort makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item "bubbles" up to the location where it belongs.

Runtime (big O): https://en.wikipedia.org/wiki/Sorting_algorithm

```
def shortBubbleSort(alist):

    exchanges = True

    passnum = len(alist)-1

    while passnum > 0 and exchanges:

       exchanges = False

       for i in range(passnum):

           if alist[i]>alist[i+1]:

               exchanges = True

               temp = alist[i]

               alist[i] = alist[i+1]

               alist[i+1] = temp

       passnum = passnum-1


alist=[20,15,66,88,55,5,77,8,3,111]

shortBubbleSort(alist)

print(alist)
```

Question 6. The selection sort improves on the bubble sort by making only one exchange for every pass through the list. In order to do this, a selection sort looks for the largest value as it makes a pass and, after completing the pass, places it in the proper location. As with a bubble sort, after the first pass, the largest item is in the correct place. After the second pass, the next largest is in place. This process continues and requires n−1 passes to sort n items, since the final item must be in place after the (n−1) st pass.




```
def selectionSort(alist):

   for fillslot in range(len(alist)-1,0,-1):

       positionOfMax=0

       for location in range(1,fillslot+1):

           if alist[location]>alist[positionOfMax]:

               positionOfMax = location


       temp = alist[fillslot]

       alist[fillslot] = alist[positionOfMax]

       alist[positionOfMax] = temp


alist = [54,26,93,17,77,31,44,55,20]

selectionSort(alist)

print(alist)
```

Question 7. The insertion sort, although still O(n2), works in a slightly different way. It always maintains a sorted sublist in the lower positions of the list. Each new item is then "inserted" back into the previous sublist such that the sorted sublist is one item larger. Figure 4 shows the insertion sorting process. The shaded items represent the ordered sublists as the algorithm makes each pass.

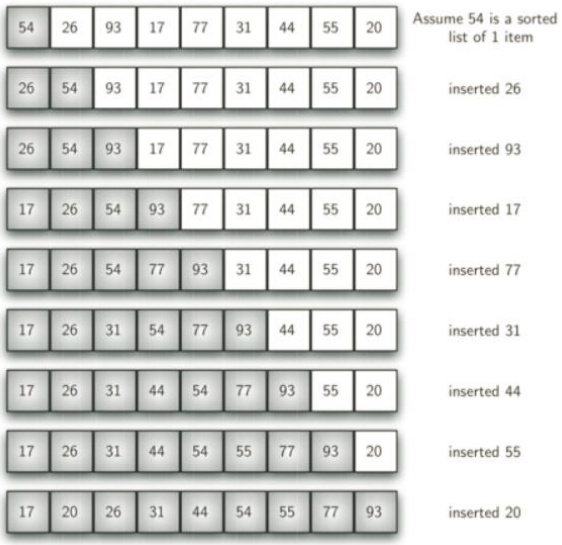
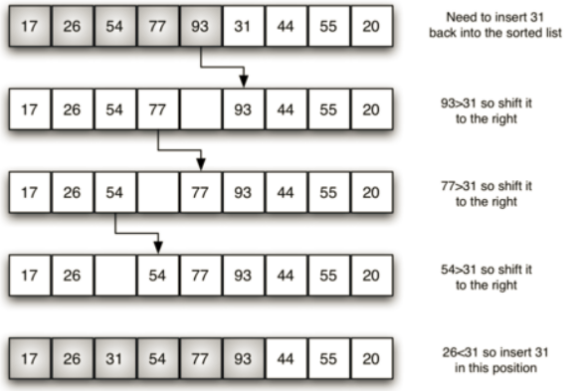


Figure below shows the fifth pass in detail. At this point in the algorithm, a sorted sublist of five items consisting of 17, 26, 54, 77, and 93 exists. We want to insert 31 back into the already sorted items. The first comparison against 93 causes 93 to be shifted to the right. 77 and 54 are also shifted. When the item 26 is encountered, the shifting process stops and 31 is placed in the open position. Now we have a sorted sublist of six items.




```
def insertionSort(alist):

   for index in range(1,len(alist)):
```

```
    currentvalue = alist[index]

    position = index


    while position>0 and alist[position-1]>currentvalue:

        alist[position]=alist[position-1]

        position = position-1


    alist[position]=currentvalue


alist = [54,26,93,17,77,31,44,55,20]

insertionSort(alist)

print(alist)
```

Reference for more sorting algorithms: http://interactivepython.org/runestone/static/pythonds/SortSearch/toctree.html

Question 8. Calculate Factorial: The factorial of a number is the product of all the integers from 1 to that number. For example, the factorial of 6 (denoted as 6!) is 1*2*3*4*5*6 = 720. Factorial is not defined for negative numbers and the factorial of zero is one, 0! = 1.

```
# Python program to find the factorial of a number provided by the user.


# change the value for a different result

num = 7


# uncomment to take input from the user

#num = int(input("Enter a number: "))


factorial = 1
```

```
# check if the number is negative, positive or zero
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
       factorial = factorial*i
   print("The factorial of",num,"is",factorial)
```

Question 9. Check Leap Year A leap year is exactly divisible by 4 except for century years (years ending with 00). The century year is a leap year only if it is perfectly divisible by 400. For example,

2017 is not a leap year

1900 is a not leap year

2012 is a leap year

2000 is a leap year

```
# Python program to check if the input year is a leap year or not


year = 2000


# To get year (integer input) from the user
# year = int(input("Enter a year: "))


if (year % 4) == 0:
   if (year % 100) == 0:
       if (year % 400) == 0:
           print("{0} is a leap year".format(year))
```

```
        else:
            print("{0} is not a leap year".format(year))
    else:
        print("{0} is a leap year".format(year))
else:
    print("{0} is not a leap year".format(year))
```

Question 10. A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8....
The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means to say the nth term is the sum of (n-1)th and (n-2)th term.

```
# Program to display the Fibonacci sequence up to n-th term where n is provided by
the user


# change this value for a different result
nterms = 10


# uncomment to take input from the user
#nterms = int(input("How many terms? "))


# first two terms
n1 = 0
n2 = 1
count = 0


# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
```

```
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence upto",nterms,":")
   while count < nterms:
       print(n1,end=' , ')
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

Question 11. The least common multiple (L.C.M.) of two numbers is the smallest positive integer that is perfectly divisible by the two given numbers.
For example, the L.C.M. of 12 and 14 is 84.

```
# Python Program to find the L.C.M. of two input number


# define a function
def lcm(x, y):
   """This function takes two
   integers and returns the L.C.M."""


   # choose the greater number
   if x > y:
       greater = x
   else:
       greater = y


   while(True):
```

```
        if((greater % x == 0) and (greater % y == 0)):
            lcm = greater
            break
        greater += 1

    return lcm

# change the values of num1 and num2 for a different result
num1 = 54
num2 = 24

# uncomment the following lines to take input from the user
#num1 = int(input("Enter first number: "))
#num2 = int(input("Enter second number: "))

print("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2))
```

Question 12. Find All Factors of a Number:

```
# Python Program to find the factors of a number

# define a function
def print_factors(x):
    # This function takes a number and prints the factors

    print("The factors of",x,"are:")
    for i in range(1, x + 1):
        if x % i == 0:
            print(i)
```

```
# change this value for a different result.
num = 320


# uncomment the following line to take input from the user
#num = int(input("Enter a number: "))


print_factors(num)
```