Even-odd sort

- The even-odd sort of a list that has even length permutes entries so that all the even-index entries come first followed by all the odd-indexed entries. To illustrate, suppose we have the following length-8 list:

'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'

- Here are the length-4 lists of the even-indexed entries and the odd-indexed entries:

'a'	'c'	'e'	'g'

'b'	'd'	'f'	'h'

- And here is the even-odd sort of the above length-8 list:

'a'	'c'	'e'	'g'	'b'	'd'	'f'	'h'

Even-odd sort

- This operation could — but for this question you are not allowed to do so — can be carried out very simply using list slicing and list concatenation: indeed, if x has length n and n is even, then the list x[0:n:2] + x[1:n:2] is the even-odd sort of x. Implement the following procedure so that it performs as specified, using just for-loops and subscripting. No list slicing or list concatenation allowed.

- def EvenOddSort(x):

```
    """ Performs an even-odd sort of x

    Precondition: x is a list with even length"""
```

- Note that EvenOddSort does not return any values. Again, no list slicing or list concatenation allowed.

Even-odd sort (Solution)

Even-odd sort (Solution)

EvenOddSort-2

- Assuming that the procedure EvenOddSort is available, implement the following function so that it performs as specified:

```
def MultipleSort(x,N):
    """ Returns a list obtained by performing N even-odd sorts of the
    list x. The list x is not altered.

    Precondition: x is a list with   even length and N is a positive int. """
```

- Use a loop that calls EvenOddSort N times. (Don’t try to do some fancy “if N is even, I’ll get the same list back” type of reasoning.)

EvenOddSort-2 (solution)

Farthest Point (I)

Assume the existence of the following class, and that the command import math has been included before- hand.

```
class Point:
    """ Attributes:
    x       the x-coordinate    [float]
    y       the y-coordinate    [float]

    """
def __init__(self,x,y):
     self.x = x
     self.y = y

def Dist(self,other):
    """ Returns a float that is the distance from self to other.

    Precondition: other is a Point """
    return math.sqrt((self.x-other.x)**2+(self.y-other.y)**2)
```

Farthest Point (II)

- Complete the following function so that it performs as specified

```
def FarthestPt(L,idx,P)
    """ Returns an integer j with the property that the distance from
    L[j] to P is maximum among all the ***unvisited*** points.

    If idx[i] = 1, then we say that L[i] has been visited. If idx[i] = 0, then
    we say that L[i] is unvisited.

    Preconditions: L is a list of references to Point objects, P is a
    reference to a point object, and idx is a list of ints that are either
    zero or 1. The lists idx and L have the same length and idx has at
    least one zero entry. """
```

Farthest Point (Solution)

Nested Loops

- What is the output if the following is executed?

```
s = "abcd"
for i in range(4):
    for j in range(i+1,4):
        print (i, j, s[i]+s[j])
```

Nested Loops (Solution)

Dictionary

- For each key in dictionary D, write down the key and corresponding value in D.

```
D1 = {'a':'one', 'b':'two', 'c': 'three', 'd':'four’}
D2 = {'c':'five', 'd':'six', 'e': 'seven' ,'f':'eight’}
D  =  {}
for d in D1:
    D[d] = D1[d]
for d in D2:
    D[d] = D2[d]
```

Dictionary (Solution)

Lists as objects-1

- If the following is executed, then what are the first five lines of output?

```
x = [10,20,30]
for k in range(1000):
    print ("k:", k, "x in the loop", x)
    x.append(x[0])
    x = x[1:4]
```

Lists as objects-1 (Solution)

Lists as objects-2

- If the following is executed, then what is the output? For full credit you must also draw two state diagrams. The first should depict the situation just after the Q.x = 0 statement and the second should depict the situation just after the P = Point(7,8) statement.

```
P = Point(3,4)
Q  =  P
Q.x  =  0
print (Q.x, Q.y, P.x, P.y)
P = Point(7,8)
print (Q.x, Q.y, P.x, P.y)
```

Lists as objects-2 (Solution)

Lists as objects-3

- If the following is executed, then what is the output?

```
x = [10,20,30,40]
y  =  x
for k in range(4):
    print ("x is", x )
    print ("y is", y )
    print ("...." )
    x[k] = y[3-k]
print (x)
```

Lists as objects-3 (Solution)

Dictionaries

- Complete the following function so that it performs as specified

```
def   F(s,D):
    """ Returns True if s is a key for D and every element in D[s] is
    also a key in D. Otherwise returns False.

    Precondition: s is a nonempty string and D is a dictionary
    whose keys are strings and whose values are lists of strings.

    """
```

Dictionaries (Solution)

Methods and Lists of Objects

- Assume the availability of the following class:

```
class City:
"""
attributes:
name       the name of a city [str]
high:      the record high temeratures [length-12 list of int]
low:  the record low temperatures [length-12 list of int]
"""
def __init__(self,cityName,theHighs,theLows):
    """Returns a reference to a City object
    PreC: cityName is a string that names a city.
    theHighs is a length 12 list of ints.
    theHighs[k] is the record high for month k (Jan is month 0)
    theLows is a length 12 list of ints
    theLowss[k] is the record high for month k (Jan is month 0) """
    self.name = cityName
    self.high = theHighs
    self.low   = theLows
```

HotMonths()

- Complete the following method for the class City so that it performs as specified.

```
def HotMonths(self):
    """ Returns the number of months where the record
    high is strictly greater than 80.
    """
```

HotMonths()(Solution)

Hotter()

- Complete the following method for the class City so that it performs as specified. Your implementation must make effective use of the method above.

```
def Hotter(self,other):
    """Returns True if the city encoded in self has strictly more
    hot months than the city encoded in other.

    A month is hot if the record high for that month is > 80

    PreC: other is a city object """
```

Hotter()(Solution)

Variation()

- Complete the following method for the class City so that it performs as specified.

```
def Variation(self):
    """ Returns a length 12 list of ints whose k-th entry
    is the record high for month k minus the record low
    for month k. """
```

Variation() (Solution)

Exaggerate()

- Complete the following method for the class City so that it performs as specified.

```
def Exaggerate(self):
    """ Modifies self.high so that each entry is increased
    by 1 and modifies self.low so that each entry is
    decreased by 1.
    """
```

Exaggerate() (Solution)

Hottest()

- Complete the following function so that it performs as specified. Assume that the methods in parts (a) and (b) are available; your implementation must make effective use of them.

```
def Hottest(C):
    """ Returns an item from C that represents the city that
    has the most hot months.
    PreC: C is a list of references to City objects """
```

Hottest() (Solution)

Recursion: Reverse

- Write a recursive function reverse(lis) that returns a list that is reverse of the input list lis.

Recursion: Reverse (solution)

Recursion: Palindrome

- Write a recursive function isPalindrome(s) that returns True if the input String “s” is a palindrome.

Recursion: Palindrome (solution)

Recursion: sum of digits

- Write a Python function–sumDigits(n)--to get the sum of digits of a non-negative integer “n”.
- Examples:
  - sumDigits(345) -> 12
  - sumDigits(45) -> 9

Recursion: sum of digits (solution)

Recursion: sum of numbers in recursive lists

- Write a Python function--sumRecList(L)--to compute sum of numbers in recursive lists L.
- Example:
  - If L= [1, 2, [3,4], [5,6]]
  - Expected output: 21

Recursion: sum of numbers in recursive lists (solution)

MCQ-1

```
x = [10,20,30,40]
N = len(x)
for k in range(N):
    x[k] = x[N-k-1]
```

- What is the final value of x?
- A. [40,30,20,10]
- B. [40,30,30,40]
- C. [4,3,2,1]
- D. [3,2,1,0]
- E. None of These

MCQ-1 (keys)

MCQ-2

```
x = [10,20]
for i in range(5):
    x.extend(x)
m = len(x)
print (m)
```

What is the output?

A. None B. 10

C. 12 D. 32 E. 64

MCQ-2 (keys)

MCQ-3

```
x = [10,20,30,40]
s = 0
for v in x:
    s += v
```

- What is the value of s?

A. Error—illegal

B. 100

MCQ-3 (keys)

MCQ-4

```
s = 0
for k in range(3):
    for j in range(k,4):
        s += 1
Print(s)
```

Output?

A. 12 B. 9

C. 6 D. None of These

MCQ-4 (keys)

MCQ-5

```
x = [10,20,30,40]
y = x
x[2] = y[3]
print x[2],y[2]
```

- What is the output?

A. 40,30 B. 30,40

C. 40,40 D. None of These

MCQ-5 (keys)

MCQ-6

```
def fA(x):
      y = x[1:]
      y.append(x[0])
#main script below
z = [10,20,30]
fA(z)
print (z[0],z[1],z[2])
```

- Output?

A. 10 20 30
B. B. 20 30 10
C. C. None of these

MCQ-6 (keys)

MCQ-7

```
def fB(x):
    y = x[1:]
    y.append(x[0])
    return y
#main script
z = [10,20,30]
w = fB(z)
print (w[0],w[1],w[2])
```

- Output?

A. 10 20 30

B. 20 30 10

C. None of these

MCQ-7 (keys)

MCQ-8

```
>>> D = {'A':[1,2,3],'B':[4,5]}
>>> ???
>>> D
{'A':[1,2,3,5],'B':[4,5]}
```

- Which of these choices for ??? does the trick?

A. D[‘A’] = D[‘A’].append(B[1])
B. D[‘A’] = D[‘A’].append(D[‘B’][1])
C. D[‘A’].append(D[‘B’][1])
D. D[0].append(D[1][1])
E. None of these

MCQ-8 (keys)

MCQ-9

```
from math import sqrt
class Point1:
 def __init__(self,x,y):
     self.x = x
     self.y = y
     self.d = sqrt(x**2 + y**2)

P = Point1(3,4)
P.x = 0
print (P.d)
```

- Output?

A. 5
B. 4
C. Neither of these

MCQ-9 (keys)

MCQ-10

```
class C:
 def __init__(self,x,y):
     self.u = x
     self.v = y

A = C([1,2],[3,4])
A.u = A.v
A.u[1] = 5
print (A.v[0],A.v[1])
```

- Output?

A. 1 2
B. 3 4
C. 1 5
D. 3 5
E. None of these

MCQ-10 (keys)