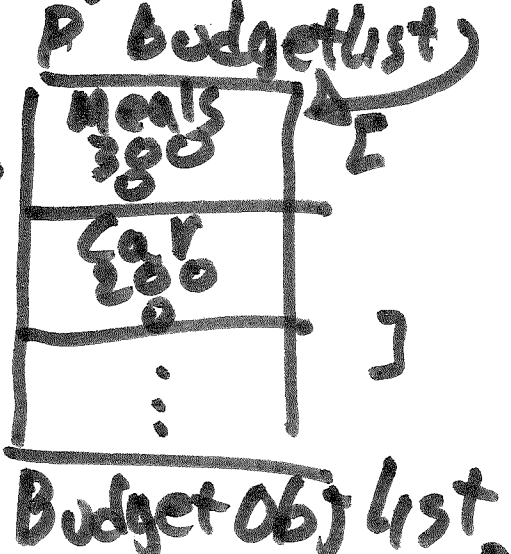
BudgetObj.py

```
class BudgetObj:
    def __init__(self, expenseType, maxAmount):
        self.exptype = expenseType
        self.maxAmount = maxAmount
        self.currentAmount = 0

    def getExpenseType(self):
        return self.exptype
    def getMaxAmount(self):
        return self.maxAmount
    def getCurrentAmount(self):
        return self.currentAmount
```

```
def setCurrentAmount(self, currentAmt):
    self.currentAmount = currentAmount
def print(self):
    print(self.expType.ljust(12),
          str("$%.0.2f" % self.maxAmount))
```




```
class BudgetObjList:
    def __init__(self, budgetFile):
        #read budgetfile and
        #add entries into BudgetObj
        #in the list.
        #read file
        f = open(budgetFile)
        lines = f.readlines()
        f.close()
```

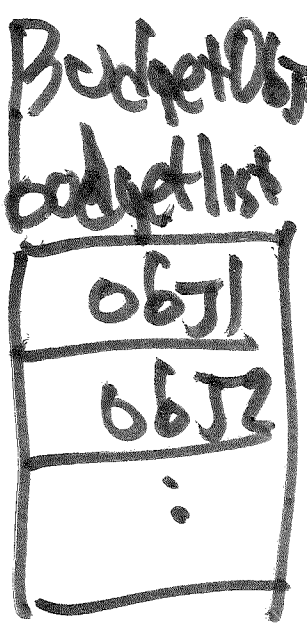
```
# Initialize budgetlist with empty list
self.budgetlist = []

#Parse the lines
for i in range(len(lines)):
    #Split fields in line
    list = lines[i].split(",")

    # Read exptype
    exptype = list[0].strip('"\n')
    if exptype == "Type":
        #Skip header of budget.txt
        continue

    #read maxamount
    maxamount = list[1].strip('$"\n')
    #Create Budget obj
    b = BudgetObj(exptype, maxamount)
```

budgetType, Maxamt
"$300.0



```
            self.budgetlist.append(b)

        #end of constructor

    def print(self): #print BudgetObjList:
        print()
        print("---=== Budget===")
        print("Type".ljust(12), "Max Amount".ljust(12))
        #print all entries in budgetlist
        total=0
        for b in self.budgetlist:
            #print this budget entry.
            b.print()
            total=total+b.getMaxAmount
        print("Total=", "$%.0.2f" % total)
def Main():
    budgetlist=BudgetObjList("budget.txt")
    budgetlist.print()
```

Call Main only if we run Budget.py directly

```
if __name__ == "__main__":
    main()
```

Step 1

Read budget.txt

~~Generate~~

readBudget(budgetFile)

return budgetList

{'exptype': exptype, 'maxAmount'}

Print type Maxamount

'School' 100

⋮

Step 2

read 'expense.txt'

readExpenses(expenseFile)

return expseList

Use budget.py as an example to write readexpense

Expenses

Print table

Date	Description	Type	Check?	Amount
02/03/2014	Indiana Water Utility	Utility		30
—	—	—		
—	—	School		

4

Step 3

Implement expensebyType

expenseByType(expenses, budget)

list of all expenses returned by readExpenses

List of all budget entries returned by readbudget

It will generate a new list expbyType that contains entries

exptype	amnt	maxamnt

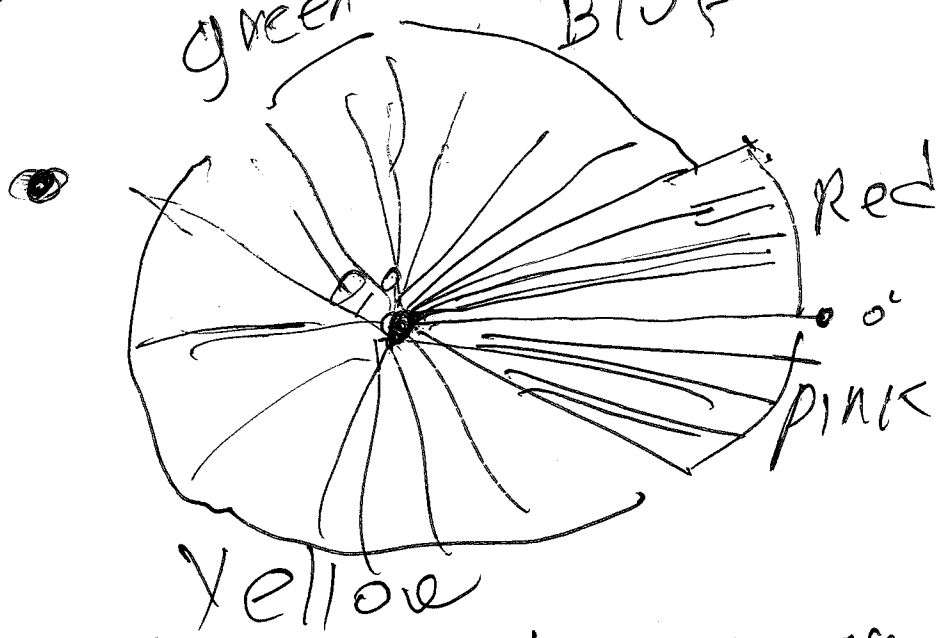
```
def expensebyType(expenses, budget):
    #initialize empty list for expByType
    expByType = [] #Initialize list with types and amnt=0
    for b in budget:
        expByType.append(
            {'exptype': b.exptype, 'amnt': 0, 'maxamnt': b.maxAmount
```

```
#Iterate over all expByType entry
for x in expByType: # utilities
    for all entries e in expenseList:
        if entry e in expense has type equal
            to x.expType.
                add amount in e to &amount in x
```

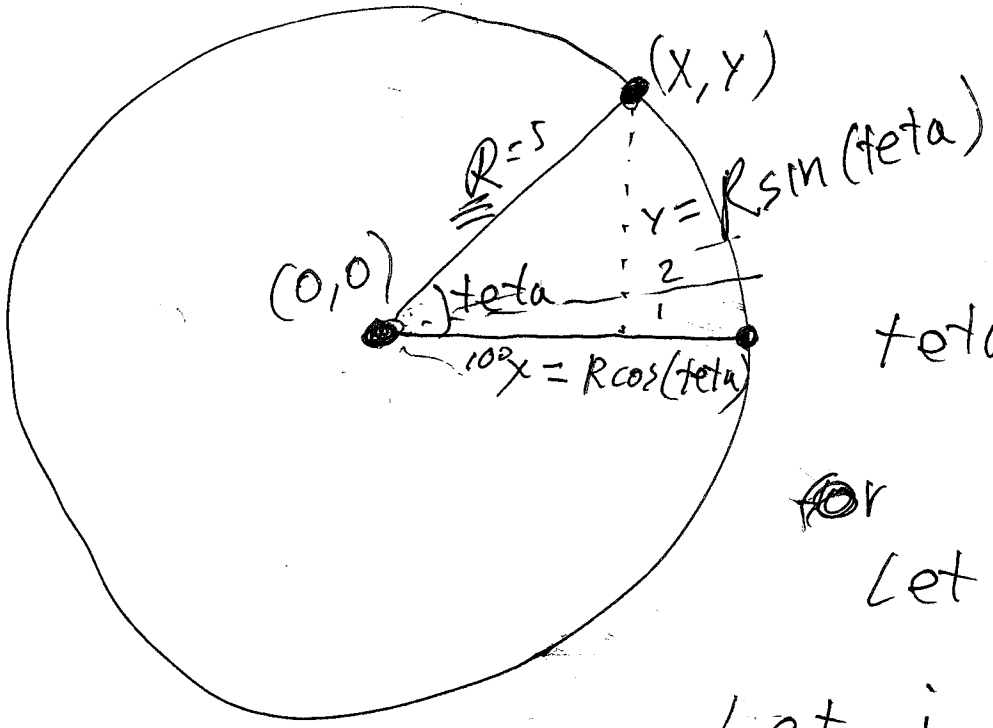
Step 4

Draw Piechart

Drawing a filled circle using lines.



Number of lines is proportional to the percentage for expense type.



teta = 0 ... 360
0 ... 2*pi
or 0 ... 1000 lines
Let n = 1000 #Number of lines

Let i the number for the ith line i = 0 ... 1000
teta = i*2*pi/n

R = 5

```
#Drawing a circle filled with lines.
n=1000
for i in range(n+1):
    teta = i*2*pi/n
    x = R*cos(teta)
    y = R*sin(teta)
```

```
#draw line
line= Line(Point(0,0), Point(X,Y))
line =setFill("red")
line =setWidth(5)
line.draw(win)
```

Ch 11. Collections

We have seen collections already in some of the programs.

Types of Collections

Lists

```
l=[]
l.append(1)
l.append(5)
l.append(7)
print("l=", l)        → Output: l=[1, 5, 7]
l[0]=3
print("l[0]=", l[0])  output: 3
print("l=", l)        → Output: l=[3, 5, 7]
```

Lists are indexed with a number that represents the position of the element in the list.

Dictionaries

```
d = { }   #initialization
d['kiwi'] = 5
d['banana'] = 8
#'Banana'
d['orange'] = 3
print("d=", d)
```

A dictionary is indexed by a "key" that often it is a string.

Output:
{'kiwi': 5, 'banana': 8, 'orange': 3}

```
print("d[kiwi]=", d['kiwi'])
```

Output: d[kiwi]= 5

```
d['banana'] = 5
print("d=", d)
```

Output:
{'kiwi': 5, 'banana': 5, 'orange': 3}

Example:

Using a dictionary to count the words in a file.

We will open the file and read it. Then we extract the words one by one and if it is the first time we find the word we add it to a dictionary with a value of 1. If it is already there we increse the value.

- We print a histogram of the words at the end.

```
#wordfreq.py
#

def main():
    print("program that prints word frequency")

    #read the name of the file
    fname = input("Type file to analyze: ")

    #Open file and read it all at once
    #and put it in string "text".
    text = open(fname,'r').read()

    text = text.lower()  # words are case insensitive

    #The text may contain special characters
    #such as punctuation like ':',',','"'..
    #we substitute them by spaces.

    #iterate over all special characters
    #and substitute by spaces
```

text:
apple banana
apple kiwi
apple
.

```
words=[ for ch in '!:;?...':
    text=text.replace(ch,'')
'apple',
'banana',
'apple',
'kiwi']#Construct a dictionary called "counts"
       #with all the occurrences of each word.
       counts={}
       for w in words:
         if w in counts:
           counts[w]=counts[w]+1 #increment
         else:                   #count for
           counts[w]=1           #this word

       #print counts in any order
       print(counts)
```

counts={
'apple':2,
'banana':1
'kiwi':1}

printing dictionary may be in any order.

```
words = text.split() # Create an array,
                     # with all words
                     # in texts
```

```
        #we would like to print
        #the counts sorted alphabetically.
        print("Counts by word alphabetically");

        #Convert the dictionary counts into a list
        #of items to be able to sort it
items:  items = list(counts.items())
[('apple',2),
 ('banana',1),
 ('kiwi',1)]

        #Now we can sort the items list
        #alphapetically
        items.sort()
        # print histogram sorted alphabetically
        for i in range(len(items)):
            word, count = items[i]  #get word and
                                    count from item
            print(word, ':', count)
```

Statistics object

- An object that stores a list of numbers and has methods that return mean, median, std deviation etc.

```
#
# statistics.py
#
class Statistics:
    def __init__(self):
        self.nums = []  #nums stores list of numbers
                        #initially empty

    def getNumbers(self): #read numbers from
                          #console
        xStr = input("Enter number (<Enter> to quit):")
        while xStr != "": #continue while input is
                          #not empty line
```

```
x = eval(xStr) #Convert xStr to
               #a number
self.nums.append(x) #add x to nums
xStr=input("Enter number<Enter>to quit):")
```

```
0  2
1  7
2  9   <- n=5//2   median=9
3  11
4  14
5
```

```
    def mean(): #compute mean (average) of nums.
        sum = 0
        for i in range(len(self.nums)):   # 0 ... n-1 = length of nums
            sum = sum + self.nums[i]
        avg = sum/len(self.nums)
        return avg
```

```
0  3
1  4
2  6   median = 6+8/2 = 7
3  8
4  10  middle = 6
5  12  middle = 3
```

```
    def median():
        self.nums.sort() # sort list of numbers.
        size= len(self.nums)
        middle= size//2
        if size % 2 != 0:
            med=self.nums[middle] #odd case
```

```
        else:
            med = (self.nums[middle] + self.nums[middle-1])
                                                    /2

        return med
```

(54)

```
    def stdDev():
        sumStdDev = 0
        m = self.mean()
        for i in range(len(self.nums)):
            dev = self.nums[i] - m
            sumStdDev = sumStdDev + dev*dev

        return sumStdDev


def main():
    s = Statistics()  #Create statistics object.
    s.getNumbers()
    m = s.mean()
    med = s.median()
```

```
    stdD = s.stdDev()
    print("mean=", m)
    print("median=", med)
    print("std Dev=", stdD)

if __name__ == '__main__':
    main()    #Only run main when
              #statistics.py is not
              # imported.
```