Ch12 Solutions Part 2

March 31, 2025

Exercise 5.1 Implement the wile statements examples for average and standard deviation using a for statement.

[2]:
```
L = [10 , -4, 4873 , -18]
total = 0
for x in L:
    total = total + x
avg = total / len (L)
print (avg )
```

```
1215.25
```

[ ]:
```
L = [10 , 20, 30, 40]
N = len (L)

total = 0
for x in L:
    total = total + x
avg = total / N

total = 0
for x in L:
    total = total + (x - avg) ** 2
12
std = ( total / N) ** 0.5
print ("Avg & Std of the list are :", avg , std )
```

```
Avg & Std of the list are :  25.0 11.180339887498949
```

Exercise 5.2 Implement the following for statement examples using a while statement.

a. Words with vowels and consonants.

b. Run-length encoding.

[ ]:
```
#   a. Words with vowels and consonants.
mixed = ["lorem","ipsum","dolor","sit","amet␣
 ↪","consectetur","adipiscing","elit",\
       ␣
 ↪"sed","do","eiusmod","tempor","incididunt","ut","labore","et","dolore","magna","aliqua"]
```

```
vowels = []
consonants = []
i = 0
N = len ( mixed )
while i<N:
    word = mixed [i]
    i += 1
    if word [0] in ['a','e','i','o','u']:
        vowels += [ word ]
    else:
        consonants += [ word ]
print ("Starting with consonant :", consonants )
print ("Starting with vowel :", vowels )
```

```
 Starting with consonant : [' lorem ', ' ipsum ', ' dolor ', ' sit ', '
consectetur ', ' adipiscing ', ' elit ', ' sed ', 'do', ' eiusmod ', ' tempor ',
' incididunt ', ' labore ', ' dolore ', 'magna ', ' aliqua ']
 Starting with vowel : ['amet ,', 'ut', 'et']
```

[ ]:
```
# b. Run-length encoding.
text ="aaaaaaxxxxmyyyaaaasssssssssttttuivvvv"
code_list = []
last_character = text [0]
count = 1
# Go over each character except for the first
N = len ( text [1:])
i = 0
while i<N:
    curr_character = text [1:][ i]
    i += 1
    # If curr_character is equal to last_character , we found a duplicate
    if last_character == curr_character:
        count += 1
    else:
    # We have finished a sequence of same characters : Save the count and
    # reinitialize last_character and count accordingly
        code_list += [ last_character if count ==1 else [ last_character ,␣
 ↪count ]]
        count = 1
        last_character = curr_character
# handle the last_character here :
code_list += [ last_character if count ==1 else [ last_character , count ]]
print ( code_list )
```

```
[' ', ['a', 6], ['x', 4], 'm', ['y', 3], ['a', 4], ['s', 9], ['t', 3], 'u', 'i',
['v', 4], ' ']
```

Exercise 5.3 Write a Python code that removes duplicate items in a list. For example, `[12, 3, 4, 12]` should be changed to `[12, 3, 4]`. The order of the items should not change.

[ ]:
```
L = list ("aaaaaaadadreeseaess")
N = len (L)
i = 0
while i<N:
    j = i+1
    while j<N:
        if L[i]==L[j]:
            del L[j]
            N -= 1
        else :
            j += 1
    i += 1

print (L)
```

```
[' ', 'a', 'd', 'r', 'e', 's']
```

Exercise 5.4 Write a program that takes a list of strings and prints the frequency of each word in the list. Hint: Make use of a dictionary. Example: For the following list,

```
["apple","banana","apple","cherry","orange","banana",
"cherry","elderberry","orange","fig","fig","fig"]
```

Your code should print the following:

```
Frequency of words:
apple: 2
banana: 2
cherry: 2
orange: 2
elderberry: 1
fig: 3
```

[ ]:
```
words = ["apple","banana","apple","cherry","orange","banana",\
        "cherry","elderberry","orange","fig","fig","fig"]

frequency = {}
for word in words :
    if word in frequency :
        frequency [ word ] += 1
    else :
        frequency [ word ] = 1

print ("Frequency of words :")
for word , count in frequency.items (): # the way to access all
    print (f"{ word }: { count }") # the dictionary element
```

```
Frequency of words :
apple: 2
banana: 2
cherry: 2
orange: 2
elderberry: 1
fig: 3
```

Exercise 5.5 You are given a dictionary of food and price information, stored under a variable Menu. Here is an example:

```
python     Menu = {"soup":22.5,"fries":15.0,"coke":10.0, "beer":30.0,
"bread":5.0,"hamburger":65.0,"scrambled egg":27.5,             "coffee":18.0,"pastry":23.5,"la
pizza":70.0,            "medium pizza":40.0,"small pizza":25.0,"salad":7.75}
```

```
Write a program that takes a list of items (from the menu), i.e., the bill as:

```python
["hamburger","fries","beer","medium pizza","beer"]
```


and then, prints the sum as follows:

```python
180.0
```


Now, modify your program so that the same bill above can be entered as:

```python
["hamburger","fries", (2,"beer"),"medium pizza"]
```
```

[ ]:
```python
Menu = {"soup": 22.5 ,"fries": 15.0 ,"cok": 10.0 ,"beer": 30.0 ,\
        "bread": 5.0 ,"hamburger": 65.0 ,"scrambled egg": 27.5 ,\
       "coffee": 18.0 ,"pastry": 23.5 ,"large pizza": 70.0 ,\
       "medium pizza": 40.0 ,"small pizza": 25.0 ,"salad": 7.75}

bill = ["hamburger","fries","beer","medium pizza","beer"]

total_price = sum ( Menu [ item ] for item in bill )
print ( total_price )
```

```
180.0
```

[ ]:
```python
Menu = {"soup": 22.5 ,"fries": 15.0 ,"cok": 10.0 ,"beer": 30.0 ,\
        "bread": 5.0 ,"hamburger": 65.0 ,"scrambled egg": 27.5 ,\
       "coffee": 18.0 ,"pastry": 23.5 ,"large pizza": 70.0 ,\
       "medium pizza": 40.0 ,"small pizza": 25.0 ,"salad": 7.75}
```

```
bill = ["hamburger","fries", (2,"beer"),"medium pizza","beer"]


total_price = sum ([ item [0]* Menu [item [1]] if type ( item ) == tuple else␣
 ↪Menu [ item ] for item in bill ])
print ( total_price )
```

```
210.0
```

Exercise 5.6 Write a rectangle that asks the user width and height and that draws a rectangle of 'x':

```
xxxxxxxxxxxx
x          x
x          x
x          x
xxxxxxxxxxxx
```

Check that only positive value (> 0) can be entered: refuse incorrect inputs and give the user the opportunity to reenter.

Tip: you can print a character without a newline with

```
print("x",end="")
```

[20]:
```
width = -1
while width < 0:
    width = int(input("Enter the width as a positive integer:"))
    if width < 0:
        print ("Please enter a positive integer")

height = -1
while height < 0:
    height = int(input("Enter the heigth as a positive integer:"))
    if height < 0:
        print ("Please enter a positive integer")

print(f"Printing a {width}x{height} rectangle:")

for i in range ( height ):
    for j in range(width):
        if j == 0 or j == width - 1 or i == 0 or i == height - 1:
            print ("*", end="")
        else:
            print (" ", end="")
    print ()
```

```
Printing a 20x5 rectangle:
********************
```

```
*                   *
*                   *
*                   *
********************
```

Exercise 7.1 Define a class `Vec3d` to represent 3 dimensional vectors. `Vec3d` should encapsulate 3 float values, `x`, `y`, and `z`. You should implement the following methods for your class: * `__init__(self, x,y,z)`: The constructor, which gets 3 values and construct a 3D vector. * `__str__(self)`: Returns the string representation of vector as `(x, y, z)` so that `print()` and `str()` functions can display the vector in a human-readable form. * `add(self, b)`: Constructs and returns a new `Vec3d` object which is the addition of the current object and `b` which is another `Vec3d` object. Vector addition is defined as the addition of all corresponding dimensions. * `len(self)`: Returns length (norm) of the vector as a scalar: $\sqrt{x^2 + y^2 + z^2}$. * `norm(self)`: Construct and return a vector in the same direction but the length is `1.0`. Simply divide all dimensions by the length of the vector.

The test run and its output would look as follows: `python a = Vec3d(1,0,0) b = Vec3d(0,1,0) c = Vec3d(0.7,0.7,0.7) print(a.add(b)) print(a.add(c)) print(c.len()) (1 , 1, 0) (1.7 , 0.7, 0.7) 1.212435565298214`

[3]:
```python
class Vec3d :
    def __init__ (self , x, y, z): # constructor
        self .x, self .y, self .z = x, y, z # initialize members from parameters
    def __str__ ( self ):
        return '({} , {}, {}) '. format ( self .x, self .y, self .z)
    def add (self , b): # construct and return in a single line
        return Vec3d ( self .x + b.x, self .y + b.y, self .z + b.z)
    def len(self):
        return ( self .x ** 2 + self .y ** 2 + self .z ** 2) ** 0.5

def main():
  a = Vec3d(1,0,0)
  b = Vec3d(0,1,0)
  c = Vec3d(0.7,0.7,0.7)
  print(a.add(b))
  print(a.add(c))
  print(c.len())

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

```
(1 , 1, 0)
(1.7 , 0.7, 0.7)
1.212435565298214
```

Exercise 7.2 Modify the Vec3d class and replace `add`with the special function `__add__()`. This change will enable the `+` operator to be used on Vec3d objects. After this change, the test run should look like this:

```
a = Vec3d(1,0,0)
b = Vec3d(0,1,0)
c = Vec3d(0.7,0.7,0.7)
print(a + b)
print(a + c)
print(c.len(), c.norm())
```

[ ]:
```
class Vec3d :
    def __init__ (self , x, y, z): # constructor
        self .x, self .y, self .z = x, y, z # initialize members from parameters
    def __str__ ( self ):
        return '({} , {}, {}) '. format ( self .x, self .y, self .z)
    def __add__ (self , b): # construct and return in a single line
        return Vec3d (self.x + b.x, self.y + b.y, self.z + b.z)
    def len(self):
        return ( self.x ** 2 + self.y ** 2 + self.z ** 2) ** 0.5

def main():
  a = Vec3d(1,0,0)
  b = Vec3d(0,1,0)
  c = Vec3d(0.7,0.7,0.7)
  print(a+b)
  print(a+c)
  print(c.len())

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

```
(1 , 1, 0)
(1.7 , 0.7, 0.7)
1.212435565298214
```

Exercise 7.3 Op een PCB gebruikt een bedrijf een schakeling van vier condensatoren in serie. De condensatoren op eenzelfde PCB moeten tot hetzelfde lot behoren.

Zoals gekend kan de vervangcapaciteit berekend worden als:

$$\frac{1}{C_t} = \frac{1}{C_1} + \frac{1}{C_2} + \frac{1}{C_3} + \frac{1}{C_4}$$

Ontwerp een object georiënteerd systeem dat van elke condensator de capaciteit en het lot bijhoudt en waarbij een PCB voorgesteld wordt als een lijst van condensatoren. Bij het samenstellen van de PCB wordt een exception gestuurd als een condensator wordt toegevoegd uit een ander lot als de reeds aanwezige condensatoren. Vang deze exception op in het main-programma.

[2]:
```
class Condensator:
    def __init__(self,capaciteit,lot):
        self.capaciteit = capaciteit
        self.lot = lot
```

```
    def __str__(self):
        print(f'capaciteit = {self.capaciteit} , lot = {self.lot}')
    def __str__(self):
        return 'capaciteit = {} , lot = {}'.format(self.capaciteit,self.lot)

class PCB:
    def __init__(self):
        self.condensators = []
        self.lot = None

    def add_condensator(self,condensator):
        if self.lot == None:
            self.lot = condensator.lot
        elif self.lot != condensator.lot:
            raise ValueError("Condensators komen uit verschillende loten")

        self.condensators.append(condensator)

    def vervang_capaciteit(self):
        for c in self.condensators:
            print(c.__str__())
        cap = 1/sum(1/c.capaciteit for c in self.condensators)
        return cap

def main():
    c1 = Condensator(10,1)
    c2 = Condensator(20,1)
    c3 = Condensator(30,1)
    c4 = Condensator(40,1)

    pcb = PCB()
    try:
        pcb.add_condensator(c1)
        pcb.add_condensator(c2)
        pcb.add_condensator(c3)
        pcb.add_condensator(c4)

        print(f'De vervangcapaciteit bedraagt: {pcb.vervang_capaciteit()}')
    except ValueError as e:
        print(e)

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

```
capaciteit = 10 , lot = 1
capaciteit = 20 , lot = 1
capaciteit = 30 , lot = 1
capaciteit = 40 , lot = 1
```

```
De vervangcapaciteit bedraagt: 4.8
```