Programming in Python
2nd Homework Solutions

Karlo Knežević
karloknezevic.github.io

November 15, 2021

1 Solutions

The solutions are intentionally given in *pdf* format with the aim of writing scripts by yourself.

1.1 Task 1

```
#this is a common trick that the user must
#enter value as long as
#does not enter the required value
while True:
        n = input ("Enter a natural number:")

        if not n.isdigit ():
                print ("You did not enter a natural number")
                continue

        n = int(n)
        break

fact = 1
for i in range (1, n + 1):
        fact *= i

print (f"{n}! = {fact}")
```

1.2 Task 2

```
import random
import math

#validation of user input
while True:
        n = int (input("Enter the number of list elements:"))

        if n> 0:
                break

#validation of user input
while True:
        min_ = int (input("Enter the lower interval limit:"))
        max_ = int (input("Enter the upper limit of the interval:"))
```

```
        if max_> min_:
                break

list_ = []
for i in range (n):
        #generating values from a given interval
        value = min_ + random.random () * (max_ - min_)
        list_.append(value)

#print list
print (list_)

#find minimum and maximum
#This is a common trick
min_ = list_[0]
max_ = list_[0]
for e in list_:
        if e <min_:
                min_ = e

if e> max_:
        max_ = e

print (f"Min: {min_:.2f}")
print (f"Max: {max_:.2f}")

#calculate the mean
sum_ = 0
for e in list_:
        sum_ += e

avg = sum_ / len(list_)
print (f"Aug: {avg:.2f}")

#calculation of standard deviation
sum_squared = 0
for e in list_:
        sum_squared += (e - avg) ** 2

std = math.sqrt (sum_squared / (len (list_) -1))
print (f"Std: {std:.2f}")

#because the original list must not be changed
#is called the sorted function!
sorted_list = sorted (list_)
if len (list_)% 2 != 0:
        median = list_[len (list_) // 2]
else:
        median = (list_[len (list_) // 2] + list_[len (list_) // 2 + 1])
            / 2

print (f"Median: {median:.2f}")

threshold = 1.5 * median
print (f"Values 50% higher than the median {median:.2f} are:", end = "")
for e in list_:
        if e> threshold:
```

```
                print (f"{e:.2f}", end = "")

print ()
print (list_)
```

1.3 Task 3

```
#validation of user input
while True:
        n = input ("Enter a natural number:")

        if not n.isdigit ():
                continue

        n = int (n)
        if n > 0:
                break

a = 0
b = 1

# 0 1 1 2 3 5 8 11
# a b
#   a b
#     a b
#       a b
#variables a and b are constantly shifted to the side
#a becomes b, and b becomes a + b
for i in range (n):
        print (f"{a} ", end = "")
        a, b = b, a + b
```

1.4 Task 4

```
#number of found and number to be examined
found = 0
counter = 2
while found < 3:
        #divisor list
        divisors = []

        for i in range (1, counter):
                #if it is a divisor, add it to the list
                if counter % i == 0:
                        divisors.append(i)

        #check the sum
        sum_ = sum(divisors)

        if sum_ == counter:
                found += 1
                print (f"A perfect number {counter} was found. Its
                    divisors are {divisors}.")

        counter += 1
```

1.5 Task 5

```
import random
import math

n = int(input("Enter␣how␣many␣points␣you␣want:"))

min_ = -10
max_ = 10

T = []
for i in range (n):
        x = random.randint(min_, max_)
        y = random.randint(min_, max_)
        T.append((x, y))

distance = 0
for i in range (n):
        distance += math.sqrt((T[i][0]-T[i-1][0])**2 + (T[i][1]-T[i
           -1][1])**2)

print (f"Distance␣is␣{distance:␣.2f}")

"""
this is an optional part, and it seems cool to me !!!
"""

#we can show which points are selected
#window will be a box, and that's a list
window = [["␣" for j in range(max_ - min_ + 2)] for i in range (max_ -
   min_ + 2)]

index = 0
for t in T:
        #the point value must be transformed
        #in the index in the window
        x, y = t[0] + max_ + 1, t[1] + max_ + 1
        window[x][y] = f"X{index}"
        index += 1

for height in range (len (window)):
        for width in range (len (window [0])):
                print (f"{window[height][width]:^5s}", end = "")
        print ()
```

1.6 Task 6

```
n = int(input("Enter␣how␣many␣Mersenne␣numbers␣you␣want:"))


found = 0
counter = 2
while found < n:

        #this is a candidate for Mersenne's number
        M = 2 ** counter - 1

        #examine if it's simple
        for i in range (2, M):
                if M % i == 0:
```

```
                        break
        else:
                #if the previous loop ended without
                #break, then it's simple!
                found += 1
                print (f"M_{counter} = {M}")

        counter += 1
```

1.7 Task 7

```
import random

m = int (input ("Enter M (number of rows) matrix dimension:"))
n = int (input("Enter N (number of columns) matrix dimension:"))
A = float (input("Enter the real number A: "))

#list comprehension
M1 = [[random.randint (0,10) for c in range (n)] for r in range (m)]
M2 = [[random.randint (0,10) for c in range (n)] for r in range (m)]

print ("M1:")
for row in range (m):
        for column in range (n):
                print (f"{M1[row][column]:3d}", end = "")
        print ()

print ("M2:")
for row in range (m):
        for column in range (n):
                print (f"{M2[row][column]:3d}", end = "")
        print ()

#initialization of the sum matrix
s = [[0 for c in range (n)] for r in range (m)]
for row in range (m):
        for column in range (n):
                s[row][column] = M1[row][column] + M2[row][column]

#initialization of the difference matrix
d = [[0 for c in range (n)] for r in range (m)]
for row in range (m):
        for column in range (n):
                d[row][column] = M1[row][column] - M2[row][column]

#initialization of the multiplication matrix
mult = [[0 for c in range (n)] for r in range (m)]
for row in range (m):
        for column in range (n):
                mult[row][column] = A * M1[row][column]

print ("Sum:")
for row in range (m):
        for column in range (n):
                print (f"{s[row][column]:3d}", end = "")
        print ()

print ("Difference:")
```

```
for row in range (m):
        for column in range (n):
                print (f"{d[row][column]:3d}", end = "")
        print ()

print ("A * M1:")
for row in range (m):
        for column in range (n):
                print (f"{mult[row][column]:8.2f}", end = "")
        print ()
```

1.8 Task 8

```
sentence = input("Enter sentence:")

#convert a string to a character list
list_ = list (sentence)
print (list_)

#create an empty dictionary
dictionary = {}
for element in list_:

        #if the character is already in the dictionary, increase the
           counter
        if element in dictionary:
                dictionary[element] += 1
        else:
                #inace notes that the character appeared by
                #first time
                dictionary[element] = 1

for character, number in dictionary.items ():
        print (f"Character {character} appears {number} times.")

#use the max function to search for the maximum
most_frequent = max(dictionary.values())

for character in dictionary:
        if dictionary [character] == most_frequent:
                print (f"Most frequent {most_frequent} character is {
                   character}.")
```

1.9 Task 9

```
import random

goal = random.randint (-20, 20)
print ("A number was generated from the interval [-20, 20].")

tries = 1
while True:
        guess = int (input("Guess the number I imagined:"))

        if guess <goal:
                print ("The number generated is greater than the
                   solution offered.")
        elif guess> goal:
```

```
                print ("The number generated is less than the offered
                    solution.")
        else:
                print (f"You guessed the number from {tries} attempts!")
                break

        tries += 1
```

1.10 Task 10

```
n = int (input("Enter a one-digit positive number:"))

sum_ = 0
number = n
for i in range (30):
        print (f"{number} ", end =" ")

        sum_ += number
        # multiply a number by 10 and add the easiest digit
        number = number * 10 + n

print ()
print (f"The sum of the previous numbers is {sum_}.")
```

1.11 Task 11

```
tax = {
        "alcohol": 0.25,
        "coffee": 0.15,
        "other": 0.2
}


products = {
        1: {"name": "Beer Kar 0.5", "price": 12.5, "tax": "alcohol"},
        2: {"name": "Espresso", "price": 7, "tax": "coffee"},
        3: {"name": "Macchiato", "price": 8, "tax": "coffee"},
        4: {"name": "Americano", "price": 9, "tax": "coffee"},
        5: {"name": "Beer 0z 0.5", "price": 12.5, "tax": "alcohol"},
        6: {"name": "Coca-Cola", "price": 10, "tax": "other"},
        7: {"name": "Fanta", "price": 10, "tax": "other"},
        8: {"name": "Sprite", "price": 10, "tax": "other"},
        9: {"name": "Juice", "price": 12.0, "tax": "other"},
        10: {"name": "Coffee", "price": 9, "tax": "coffee"}
}

width = 90
columns = 6

w = width // columns

while True:
        print ("Item codes" .center (90, "="))
        print (f"{'Code':<{w}s}{'Item':^{w * 2}s}")
        for code in products:
                print (f"{code:<{w}d}{products [code]['name']:<{w * 2}s}
                    ")
        print ("-".center(90, "-"))
```

```
        enter = input (f"{'New␣receipt␣[R|r]␣or␣Quit␣[Q|q]:':>{w␣*␣3}s}"
            )

        enter = enter.lower()
        if enter == "q":
                break

        if enter != "r":
                continue

        items = []
        while True:
                code = input (f"{'Code␣[K␣|␣k]:':>{w␣*␣3}s}")

                if code.lower() == "k":
                        break

                code = int(code)

                if code not in products:
                        print ("Unknown␣code!")
                        continue

                amount = int(input(f"{'Quantity:':>{w␣*␣3}s}"))

                items.append((code, amount))
                print (f"{'Item␣added:':>{w␣*␣3}s}{products[code]['name
                    ']}␣X␣{amount}")

        print ("Receipt".center (90, "-"))
        print (f"{'Code':^{w}s}{'Item␣name':^{w}s}{'Price':^{w}s}{'Tax
            ':^{w}s}{'Quantity':^{w}s}{'Sum':^{w}s}")

        final_price = 0
        final_tax = 0

        for item in items:
                code = item[0]
                amount = item[1]

                prod = products[code]
                total = prod["price"] * amount
                total_tax = tax[prod["tax"]]

                final_price += total
                final_tax += total * total_tax

                print(f"{code:^{w}d}{prod['name']:<{w}s}{prod['price
                    ']:^{w}.2f}{total_tax:^{w}.2f}{amount:^{w}d}{total:>{
                    w}.2f}")

        print (f"{'':{w␣*␣4}s}{'Sum:':>{w}s}{final_price:>{w}.2f}")
        print (f"{'':{w␣*␣4}s}{'Tax:':>{w}s}{final_tax:>{w}.2f}")
        print (f"{'':{w␣*␣4}s}{'Total:':>{w}s}{final_price␣+␣final_tax
            :>{w}.2f}")
```