Ämne: Matematik
Elev: Nicolas Munoz
Projekt:

SMARTKLOCKAN

Mitt projekt handlar om att skapa en “smartklocka” som kommer att ha olika användningsområden. För närvarande kan man hitta alla möjliga appar och funktioner i smartklockor. Man kan till exempel göra allt från att mäta sin puls till att ringa och skicka meddelande. De flesta smartklockor har tillgång till gps och internet som gör det lättare att uppdatera data i realtid. Jag har bestämt mig för att inte göra på detta vis. Datan som min klocka kommer att använda kommer från de olika sensorerna, den blir sen beräknad enligt en kod som löser nån formula för att visa resultatet på skärmen. Jag har delat upp mitt projekt i 4 olika problem som jag har löst på mitt eget sätt det vill säga ifrån mina kunskaper. Jag är medveten om att det finns andra lösningar men är ändå jättenöjd med resultatet.

Problem 1:
Klockan ska kunna visa tiden som finns i olika länder i världen.
Lösning:
Man behöver veta tidszonen här och tidszonen som de andra länderna har och deras skillnad (Td). Sättet att beräkna det är att subtrahera det minsta tid(Tm) från det största (Ts).

$$Td = Ts - Tm$$

Sen bestämmer den om man ska addera eller subtrahera Td tiden i landet (Tz1) det beror på om tidszonen ligger till höger eller vänster av landet som vi har som referens.

- tidszon i högersidan:

$$Tz2 = Tz1 + Td$$

- tidszon i vänstersidan:

$$Tz2 = Tz1 - Td$$

```
class Zone:
    def __init__(self,namn,tidsZon,tid): #tidsZon(-12,12) tid(0,23)
        self.namn = namn
        self.tidsZon = tidsZon
        self.tid = tid

    def time(self, zone2):
        tZ1=0
        tZ2=0          #tidsZone1 och 2
```

```
        if self.tidsZon >= zone2.tidsZon:
            tZ1 = self.tidsZon
            tZ2 = zone2.tidsZon
            Dt = tZ1 - tZ2  #diference
            zone2.tid = self.tid - Dt
        else:
            tZ1 = zone2.tidsZon
            tZ2 = self.tidsZon
            Dt = tZ1 - tZ2  #diference
            zone2.tid = self.tid + Dt

        if zone2.tid < 0:
            zone2.tid *= -1


        print("Time in",zone2.namn,"is",zone2.tid,":00")

sweeden = Zone("Sweeden",1,12)
china = Zone("China",6,0)
newYork = Zone("New York",-5,0)
sweeden.time(newYork)
sweeden.time(china)
```

```
Time in New York is 6 :00
Time in China is 17 :00
```

Problem 2:
Klockan ska ha en sensor/termometer som mäter rumstemperatur. Jag vill kunna visa i stjärnan medel temperatur och annan statistik, både dagligen och under veckan.
Lösning:
Den samlar in olika data genom dagen och beräknar den information som kan vara intressant att veta som till exempel högsta och lägsta temperaturen, medeltemperaturen eller ändra till ° Fahrenheit.

- Medeltemperatur räknas ut som nedan:

$$\overline{X} = \frac{t1 + t2 + t3 + \ldots tn}{n}$$

- För att få det högsta och lägsta värdet kommer programmet att organisera alla värden och sen ta den första v[0] och den sista v[n-1]
- För att ändra från grader celsius °C till fahrenheit °F måste man applicera denna formel som finns nedan:

$$°F = \frac{°C \times 9}{5} + 32$$

- Klockan visar också denna statistiska beräkningen för hela veckan. För att kunna göra detta har jag skapat ett objekt till varje veckodag, länkar dem tillsammans för att kunna ha tillgång till dess data, och sen beräknas informationen på samma sätt som innan.
- Programmet avrundar resultaten till 1 eller 2 decimalbråk. För detta man använder funktionen round() i python.

```
class Dag:
    def __init__(self,veckadag):
        self.veckaDag = veckadag
        self.temp = []
        self.max = 0
        self.min = 0
        self.mT = 0
        self.maxFaren = 0
        self.minFaren = 0
        self.mTfaren = 0

        next = None

    def maxMin(self):
        nummerE = len(self.temp)
        for i in range(0,nummerE -1):
            for j in range (0,nummerE-i-1):
                if self.temp[j]> self.temp[j+1]:
                                            self.temp[j],self.temp[j+1]   =
self.temp[j+1],self.temp[j]

        self.max = self.temp[nummerE-1] #max temp
        self.min = self.temp[0]         #min temp
        print("Max temp:",self.max,"°C")
        print("Min temp",self.min,"°C")

    def medelTemp(self):
        nummerE = len(self.temp)
        total= 0
        for i in range(0, nummerE):
            total += self.temp[i]
        self.mT = total/nummerE
```

```
                print("Medel  temp",round(self.mT,1),"°C")  #medel  temp
avrundning
        print("")

    def fahrenheit(self):
        self.maxFaren = (self.max * 9/5) +32  #till fahrenheit
        self.minFaren = (self.min * 9/5) +32
        self.mTfaren = (self.mT *9/5) +32

        print("Max temp",self.maxFaren,"°F")
        print("Min temp",self.minFaren,"°F")
        print("Medel temp",self.mTfaren,"°F")
        print("")

class Veckan:
    def __init__(self):
        self.head = None
        self.current = None
        self.vMax = 0
        self.vMin = 0
        self.total = 0 #total element
        self.medelTV =0

    def insert(self,dag):
        if dag.veckaDag == 1:
            self.head = dag
            self.current = dag
            self.head.next = None
            self.total += 1

        else:
            self.current.next = dag
            self.current = dag
            self.current.next = None
            self.total += 1

    def maxMinTempV(self):
        i = self.head
        self.maxV = i.max
        self.minV = i.min
        while (i.next != None):
            if i.max < i.next.max:
                self.maxV = i.next.max
```

```
            if i.min > i.next.min:
                self.minV = i.next.min
            i = i.next
        print("Veckans max temp:", self.maxV, "°C")
        print("Veckans min temp:", self.minV, "°C")

    def medelV(self):
        i = self.head
        totalTemp = 0
        while (i != None):
            totalTemp += i.mT
            i = i.next
        self.medelTV = totalTemp/self.total
        print("Veckans medel temp:",round(self.medelTV,1),"°C")



monday = Dag(1)
monday.temp.append(25)
monday.temp.append(20)
monday.temp.append(22)
monday.temp.append(18)
monday.maxMin()
monday.medelTemp()
monday.fahrenheit()

tuesday = Dag(2)
tuesday.temp.append(22)
tuesday.temp.append(15)
tuesday.temp.append(12)
tuesday.maxMin()
tuesday.medelTemp()

newVeckan = Veckan()
newVeckan.insert(monday)
newVeckan.insert(tuesday)

newVeckan.maxMinTempV()
newVeckan.medelV()
```

Output:

```
Max temp: 25 °C
Min temp 18 °C
Medel temp 21.2 °C

Max temp 77.0 °F
Min temp 64.4 °F
Medel temp 70.25 °F

Max temp: 22 °C
Min temp 12 °C
Medel temp 16.3 °C

Veckans max temp: 25 °C
Veckans min temp: 12 °C
Veckans medel temp: 18.8 °C
```

Problem 3:
Jag vill att klockan ska visa farten som användaren har när hen promenerar eller springer. Sen vill jag också beräkna den totala distansen som man går och en prognos för hur många kalorier som man använder.
Lösning:
Det skulle funka med att ha någon typ av gps system som skulle beräkna distansen som man flyttar sig delat med tid men jag har tänkt att i stället kommer klockan att ha en rörelsesensor som kommer att öka med varje steg.

- **Farten**: Man behöver att definiera hur lång är ens eget steg(S). Då kommer vi att beräkna hur många steg (NS) vi har gjort. Sen multiplicerar vi båda värdena och delar resultatet med 100 så vi får distansen (D) som man har rört sig i meter i stället för cm.

$$D = \frac{NS \times S}{100}$$

Då ska man dela distansen (D) med tiden (T) som vi har satt i 60 sekunder för att uppdatera farten varje minut. Med detta får vi hur många meter per sekund vi går.

$$V = \frac{D}{T}$$

För att få en tydligare bild av farten har vi omvandlat V, i m/s, till Km/h genom att multiplicera resultatet med 3,6.

$$V = \frac{m}{s} \times 3,6 = \frac{km}{h}$$

- För att beräkna **distans** (D) kommer programmet i en variabel att samla alla steg (TS) man går, multiplicera detta med stegstorlek (S) och omvandlar det till km.

$$D = \frac{TS \times S}{100}$$

- Det finns olika sätt att få en **prognos** över hur många **kalorier** (Cal) man använder när man rör sig. Jag har använt en som är baserad på distansen (td) som man har gått. Andra värden som man behöver veta är medelfart och vikt (m).
  Medelfarten behöver man för att determinera ungefär hur många kalorier man har använt (ac) det beror på farten, om farten är större då blir det mer kalorier. Då kan vi multiplicera detta med vikten och distansen som man gick för att få prognosen.

$$\text{Cal} = ac \times m \times td$$

- Här har jag också avrundat resultaten till 1 eller 2 decimalbråk med funktionen round() i python.

```
class Farten:
    def __init__(self,stepSize):
        self.stepS = stepSize #how long is user step i cm
        self.nSteps = 0  #total steps
        self.tid = 60  #sekunder
        self.walkTid =  0.017 # total tid timer
        self.fartms = 0
        self.fartkm = 0
        self.totalDistKm = 0

    def farten(self, steps):
        self.nSteps += steps
        distans = self.stepS * steps /100 # omvandling till meter
        self.fartms = distans / self.tid  # farten, meter per sekund
        self.fartkh = self.fartms * 3.6  # omvandling km/h


        print("Fart:",round(self.fartkh,2),"km/h")


    def distans(self):
        totalDistM = (self.nSteps * self.stepS)/100 # meter
```

```
        self.totalDistKm = totalDistM/ 1000 # km

        print("Distans:",round(self.totalDistKm,2),"Km")
        print("Distans:",round(totalDistM,2),"m")

    def kalorier(self,vikt):
        medelFart = self.totalDistKm/self.walkTid

        if (medelFart< 4 ):
            cal = 0.48* vikt* self.totalDistKm
        elif(medelFart < 7):
            cal = 0.73* vikt* self.totalDistKm
        else:
            cal = 1.03 * vikt* self.totalDistKm
        print("Kalori användning:", round(cal,2),"cal")




stepSize = 80 #cm
nummerSteps = 90 # indata varje 60 sekunder
vikt = 60 #kg
f1= Farten(stepSize)
f1.farten(nummerSteps)
f1.distans()
f1.kalorier(vikt)
```

Output:

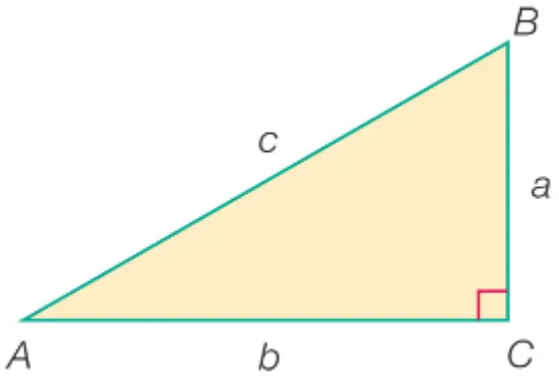
```
Fart: 4.32 km/h
Distans: 0.07 Km
Distans: 72.0 m
Kalori användning: 3.15 cal
```

Problem 4:

Att klockan kan beräkna väglutning kan vara en till intressant egenskap för de som vill använda det.

Lösning:

- För att kunna beräkna väglutning man kan hjälpa sig av trigonometri och mer konkret formulan:



$$C^2 = a^2 + b^2$$

Vi har redan ett sätt för att kunna beräkna **hypotenusa** (**c**) som jag kommer att interpretera det som distansen som finns mellan 2 punkter. Då behöver vi veta en av de andra sidor som en triangel rektangel har. I detta fall ska vi börja att beräkna sidan **a** som kommer att motsvara triangelns **höjd**.

- För att veta höjden borde klockan ha en till sensor, i detta fall en **barometer** som är ett instrument som kommer att mäta lufttrycket i **pascal** (Pa). Skillnaden på lufttrycket mellan dessa 2 punkter ska tillåta oss att beräkna höjden med formulan nedanför:

$$h = \frac{\Delta p}{çA \times g}$$

Där h = höjd , Δp = skillnad mellan trycket, $çA$ = luft densitet (1,225 kg/m^3) och g = gravitation (9,8 m/s).
Nu när vi hari **a** (hojd) kan vi få **b** som kommer att bli:

$$b^2 = c^2 - a^2$$

- Sista steget är att applicera formulan för att kunna beräkna lutningen som är:

$$\%\ lutning = \frac{a}{b} \times 100$$

Så får vi som resultat procent lutning som väggen har.

- Om vi vill veta hur många grader lutningen har då kan vi använda formulan:

$$sen\ A = \frac{a}{c}$$

Där A (angle) kommer att bli **arcsin** av **a** delat med **c**.