Python Programming and Data Analysis

Bin Han

2025-09-08

Table of contents

1 Project Description

The collection of programming assignments presented covers a wide range of fundamental Python programming concepts, data handling, and visualization techniques. The initial tasks, such as counting words and letters in a sentence, constructing a mirrored staircase of numbers, and implementing temperature converters, focus on developing strong skills in string manipulation, loops, conditional statements, and function design. More advanced tasks like FizzBuzz reinforce the use of modular arithmetic, iteration, and user interaction to build adaptable algorithms. The birthday problem introduces the datetime module, strengthening understanding of working with real-world date and time data and formatting output in a user-friendly way.

The later assignments expand into applied programming with data visualization and analysis. The stock savings simulator leverages matplotlib.pyplot to model compound growth and demonstrate the impact of annual deposits and interest rates over time, providing experience in numerical simulation and graphical presentation. The fuel price analysis further deepens data-handling skills through reading and parsing CSV files with the csv module, manual computation of statistical measures, and visualization of price trends across time. Together, these exercises foster not only problem-solving and algorithmic thinking, but also practical skills in importing and applying standard Python libraries for data analysis, visualization, and user-driven interactivity—key foundations for real-world programming and data science applications.

2 Minsta och Största

Skriv ett Python-program som ber användaren skriva in en mening. Programmet ska sedan skriva ut:

1. Hur många ord meningen innehåller
2. Hur många bokstäver den innehåller (mellanslag ska inte räknas som bokstäver)

```
print("skriv en mening:")
mening="Jag gillar att programmera i Python!"
print(mening)

word_list=mening.split()#Dela upp mening till ord
word_num=len(word_list)#Antal ordet
num=0
for item in word_list:
    num=num+len(item)#Antal bokstäver
print(f"Antal ord: {word_num}")
print(f"Antal bokstäver (utan mellanslag): {num}")
```

```
skriv en mening:
Jag gillar att programmera i Python!
Antal ord: 6
Antal bokstäver (utan mellanslag): 31
```

3 Spegelvänd trappa

Skriv ett Python-program som:

Beskrivning

Skriv ett program som frågar användaren efter ett heltal x och sedan skriver ut en "spegelvänd" trappa med x rader. På varje rad visas stigande tal från 1 upp till radnumret, högerjusterat.

```
# Tentamen 2025-08-18 -- 19

# Skriv din kod nedan
print("Välj ett heltal:")
num=5
for i in range(1,num+1):# loop each row
    list1=[]
    for j in range(1, i+1):# loop each element as string of each row
        list1.append(str(j))
    string=" ".join(list1)# join each element of each row into one string
    print(string.rjust(num*2-1)) # print out each row with right aligns.
```

```
Välj ett heltal:
        1
      1 2
    1 2 3
  1 2 3 4
1 2 3 4 5
```

4 Temperaturomvandlare

Skriv ett Python-program som:

Beskrivning

Du ska skapa ett program som innehåller två funktioner:

Omvandla Celsius till Fahrenheit Omvandla Fahrenheit till Celsius 2 Du behöver alltså

Fråga användaren om hen vill omvandla från Celsius till Fahrenheit eller tvärtom

Be om en temperatur att omvandla

Anropa rätt funktion och visa resultatet

Fråga om användaren vill göra en ny omvandling – programmet fortsätter tills användaren tackar nej

OBS!

En °C motsvarar 1,8 °F (1,8 kan räknas som 9/5) 0 °C är lika med 32 °F

```python
def CeltillFahren(Celsius):
    Fahren= Celsius*9/5+32
    print(f"{Celsius}°C är {Fahren}°F")
def FahrentillCel(Fahren):
    Celsius=(Fahren-32)*5/9
    print(f"{Fahren}°F är {Celsius}°C")



def menu():
        Num=print("Vill du omvandla (1) Celsius till Fahrenheit eller (2) Fahrenheit till
        print("Ange temperaturen i Celsius: 20")
        CeltillFahren(20)
        print("Vill du göra en ny omvandling? (j/n): j")
        Num=print("Vill du omvandla (1) Celsius till Fahrenheit eller (2) Fahrenheit till
        print("Ange temperaturen i Fahrenheit: 50")
        FahrentillCel(50)
        print("Vill du göra en ny omvandling? (j/n): n")
        print("Tack för att du använde temperaturomvandlaren!")

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

```
Vill du omvandla (1) Celsius till Fahrenheit eller (2) Fahrenheit till Celsius?: 1
Ange temperaturen i Celsius: 20
20°C är 68.0°F
Vill du göra en ny omvandling? (j/n): j
Vill du omvandla (1) Celsius till Fahrenheit eller (2) Fahrenheit till Celsius?: 2
Ange temperaturen i Fahrenheit: 50
50°F är 10.0°C
Vill du göra en ny omvandling? (j/n): n
Tack för att du använde temperaturomvandlaren!
```

5 Fizz Buzz

Skriv ett Python-program som:

Beskrivning

Skriv ett program som frågar användaren efter ett slutvärde n. Programmet ska sedan skriva ut talen från 1 till n. Användaren ska sedan ange två tal: Fizz och Buzz.

Frågar användaren efter ett slutvärde (n).

Frågar användaren vilket tal som ska ge “Fizz”.

Frågar vilket tal som ska ge “Buzz”.

Sedan skriver ut talen från 1 till n enligt reglerna:

Om talet är delbart med “Fizz”-talet → skriv “Fizz”

Om talet är delbart med “Buzz”-talet → skriv “Buzz”

Om talet är delbart med båda → skriv “FizzBuzz”

Annars → skriv ut själva talet

```
# Skriv din kod nedan
Slutvärde=10
Fizzvärde= 2
Buzzvärde=5

for i in range(1, Slutvärde+1):
    if i % Fizzvärde==0 and i % Buzzvärde ==0:
        print("FizzBuzz")
    elif i % Fizzvärde ==0:
        print("Fizz")
    elif i % Buzzvärde ==0:
        print("Buzz")
    else:
        print(i)
```

```
1
Fizz
3
Fizz
Buzz
Fizz
7
Fizz
9
FizzBuzz
```

6 Födelsedag-dag

Inga moduler förutom modulen datetime får importeras till ditt program. Om någon ytterligare modul importeras får du 0 poäng på uppgiften.

Skriv ett Python-program som:

Beskrivning

Du ska ta reda på vilken veckodag du föddes på!

Skriv ett program som:

Frågar användaren efter sitt födelseår, månad och dag (som tre separata inmatningar).

Använder modulen datetime för att skapa ett datumobjekt.

Tar reda på vilken veckodag användaren föddes.

Skriver ut detta med veckodagens namn, t.ex. “Du föddes på en måndag.” (dagen får inte skrivas ut på engelska)

```
import datetime

år=1990
månad= 5
dag= 14

# create a dateobject
födelsedatum = datetime.date(år, månad, dag)

# pick right weekday
veckodagar= ["måndag", "tisdag", "onsdag", "torsdag", "fredag", "lördag", "söndag"]

veckodag= veckodagar[födelsedatum.weekday()]

print(f"Du föddes på en {veckodag}")
```

```
Du föddes på en måndag
```

7 Aktiesparande över tid med årlig insättning

Inga moduler förutom modulen matplotlib.pyplot får importeras till ditt program. Om någon ytterligare modul importeras får du 0 poäng på uppgiften.

Skriv ett Python-program som:

Beskrivning

Du har börjat spara i en aktie som stiger varje år. Nu vill du simulera hur mycket ditt sparande kan växa över tid, beroende på hur mycket du själv sätter in varje år.

Skriv ett program som:

Frågar användaren:

Hur många år du vill spara.

Hur stor den årliga insättningen är (t.ex. 5000 kr).

Hur många procent aktien stiger per år (t.ex. 7 %).

Räknar hur kapitalet växer år för år:

Varje år lägger du till den insättning du valt.

Sedan växer kapitalet med angiven procent.

Använder matplotlib.pyplot för att rita ett linjediagram som visar utvecklingen.

```
import matplotlib.pyplot as plt

sparaår= 15
sparavarje=5000
aktien=7

print(f"Hur många år vill du spara? {sparaår}")
print(f"Hur mycket vill du spara varje år? {5000}")
print(f"Hur många procent stiger aktien per år? {aktien}")

list1= []
for i in range(1, sparaår+1):
    list1.append(i)

list2= []
sum=0
sum=sparavarje
for i in range(sparaår):
    sum=sum*(1+aktien)
    sum=sum+sparavarje
    list2.append(sum)

plt.plot(list1, list2, marker ="o")
plt.xlabel("År")
plt.ylabel("Totalt värde (kr)")
plt.title("Utveckling av aktiesparande")
plt.grid(True)
plt.show()
```

```
Hur många år vill du spara? 15
Hur mycket vill du spara varje år? 5000
Hur många procent stiger aktien per år? 7
```

8 Drivmedelsanalys

Inga moduler förutom modulerna csv och matplotlib.pyplot får importeras till ditt program. Om någon ytterligare modul importeras får du 0 poäng på uppgiften.

Skriv ett Python-program som:

Beskrivning

I skenet av att drivmedelspriserna har ökat betydligt under den senaste tiden ska du i denna uppgift analysera och presentera prisutvecklingen på bensin, E85 och diesel under perioden 2015-01-01 – 2021-10-15. Prisuppgifterna är hämtade från OKQ8:s hemsida (https://www.okq8.se) och finns sparade i csv-filen fuelprices_day_by_day.csv Ladda ner fuelprices_day_by_day.csvNedanstående figur visar delar av innehållet i denna fil.

Tips på hur du går till väga för att lösa uppgift 7:

Öppna filen fuelprices_day_by_day.csv Ladda ner fuelprices_day_by_day.csvi Excel eller i något annat kalkylprogram och bekanta dig med innehållet. Läs därefter noga igenom de tre deluppgifterna (7.a, 7.b och 7c) så att du får en god uppfattning om vad respektive uppgift går ut på. Fundera därefter, per uppgift, över vilken information du behöver för att lösa uppgiften. Överväg att skissa upp ett enkelt flödesdiagram som visar hur du tänker lösa respektive uppgift. Ofta är det tidssparande att göra en sådan analys innan man startar kodningen.

8.1 Uppgift 7a

Skriv ett program som läser in (kopierar) csv-filen fuelprices_day_by_day.csv Ladda ner fuelprices_day_by_day.csvtill en tvådimensionell lista med namnet fuelprices. Separationstecknet är semikolon (;).

Kontrollera att listan fuelprices är en kopia på innehållet i csv-filen fuelprices_day_by_day.csv Ladda ner fuelprices_day_by_day.csvgenom att skriva ut de första fem raderna i listan fuelprices.

```
import csv
import matplotlib.pyplot as plt

fuelprices = []

with open("fuelprices_day_by_day.csv", "r", encoding="utf-8") as f:
    reader = csv.reader(f, delimiter=";") #
    for row in reader:
        fuelprices.append(row)#create the list where each item is each list of row.

for row in fuelprices[:5]:
    print(row)
```

```
['Datum', 'Bensin 95', 'E85', 'Diesel']
['2015-01-01', '12.59', '9.5', '12.85']
['2015-01-02', '12.59', '9.49', '12.95']
['2015-01-03', '12.59', '9.49', '12.95']
['2015-01-04', '12.59', '9.49', '12.95']
```

8.2 Uppgift 7b

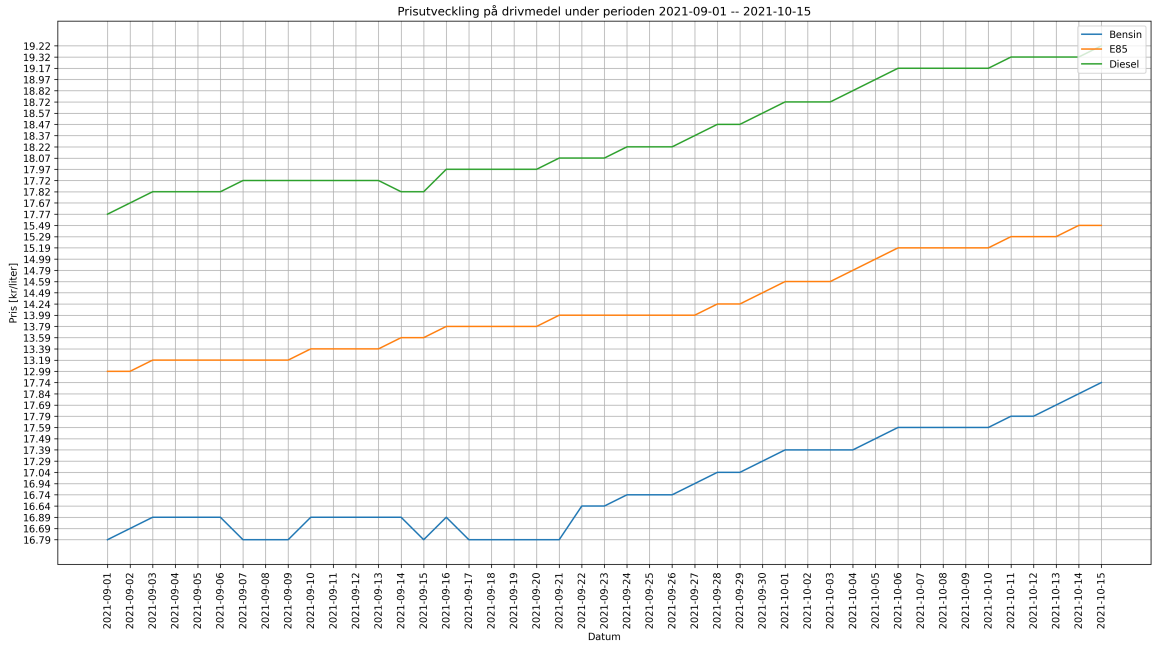
I denna uppgift ska du plotta priset för bensin, E85 och Diesel under perioden 2021-09-01 – 2021-10-15 i samma diagram. Diagrammet ska ha en lämplig titel och det ska finnas förklarande text på axlarna. Det ska också finnas förklarande text (sk etikett) till varje kurva och diagrammet ska ha ett rutnät.

Förtydligande:

Endast modulen pyplot får användas i denna uppgift. Tips: Du kan infoga programsatsen figure(figsize=(20,10)) för att förstora diagrammet och programsatsen plt.xticks(rotation=90) för att vrida värdena på x-axeln 90 . Nedan visas ett förslag på ett lämpligt diagramutseende där graferna har utelämnats.

```
print("Uppgift 7b (5p)\n")
# Skriv din kod nedan
# Create the list we need
Timlist=[]
Benlist=[]
E85list=[]
Dielist=[]

start= "2021-09-01"
end="2021-10-15"

for i in range(len(fuelprices)):
    if start <=fuelprices[i][0]<= end:
        Timlist.append(fuelprices[i][0])
        Benlist.append(fuelprices[i][1])
        E85list.append(fuelprices[i][2])
        Dielist.append(fuelprices[i][3])

# Create the plot
plt.figure(figsize=(20, 10))
plt.plot(Timlist, Benlist, label="Bensin")
plt.plot(Timlist, E85list, label="E85")
plt.plot(Timlist, Dielist, label="Diesel")

# Add labels and title
plt.xlabel("Datum")
plt.ylabel("Pris [kr/liter]")
plt.title("Prisutveckling på drivmedel under perioden 2021-09-01 -- 2021-10-15")
plt.xticks(rotation= 90)

# addlegend
plt.legend(loc="upper right")

# add grid lines
plt.grid()

# show the plot
plt.show()
```

```
Uppgift 7b (5p)
```



8.3 Uppgift 7c

I den övre delen av tabellen visas under hur lång tid av mätperioden (uttryckt i procent) som respektive drivmedel legat inom ett visst prisintervall. Datumen anger när respektive drivmedel för första gången förekommer inom prisintervallet. Dubbla minustecken (–) betyder att inget pris har registrerats inom prisintervallet.

Den undre delen av tabellen visar det högsta-, lägsta- och medelvärdet av priset för respektive drivmedel under hela perioden.

Din uppgift är att skriva ett program som skapar en tabell med en utformning enligt tabellen i Tabell_uppg_7.pdf Ladda ner Tabell_uppg_7.pdfsom innehåller korrekt information baserat på innehållet i listan fuelprices.

Förtydligande:

Endast modulen csv får användas i denna uppgift. Det är inte heller tillåtet att använda funktionerna min(), max() och sum() för att beräkna lägsta-, högsta- och medelpriset. Kursen innehåller inget moment som behandlar formatering av tabeller. Om du därför inte lyckas med att få tabelldata i 'snygga kolumner' så gör det ingenting. Det resulterar inte i något poängavdrag. Däremot måste (givetvis) tabellinnehållet vara korrekt för att få full poäng. Du får även poäng om du bara gör delar av uppgiften, t.ex. bara beräknar min-, max- och medelvärdet. Tips: Kontrollera dina svar med avseende på min- ,max- och medelvärde genom att utföra dessa operationer på filen csv direkt i Excel eller i ett motsvarande kalkylprogram.

```
Tim2list=[]
Ben2list=[]
E852list=[]
Die2list=[]

start2= "2015-01-01"
end2="2021-10-15"

for i in range(len(fuelprices)):
    if start2 <=fuelprices[i][0]<= end2:
        Tim2list.append(fuelprices[i][0])
        Ben2list.append(fuelprices[i][1])
        E852list.append(fuelprices[i][2])
        Die2list.append(fuelprices[i][3])

# Skriv din kod nedan
def lägstapris(list1):
    list1=[float(x) for x in list1] # convert to numbers
    minvalue=list1[0]
    for i in range(1, len(list1)):
        if minvalue>list1[i]:
            minvalue=list1[i]
    return minvalue

def högstapris(list1):
    list1=[float(x) for x in list1] # convert to numbers
    maxvalue=list1[0]
    for i in range(1, len(list1)):
        if maxvalue<list1[i]:
            maxvalue=list1[i]
    return maxvalue

def Medelpris(list1):
    list1=[float(x) for x in list1] # convert to numbers
    sum=0
    for i in range(len(list1)):
        sum=sum+list1[i]
    medelpris= sum/len(list1)
    return round(medelpris, 2)

table= [
    ["" ,"Bensin [kr/liter]", "E85 [kr/liter]", "Diesel [kr/liter]"],
    ["Lägsta pris", lägstapris(Ben2list), lägstapris(E852list), lägstapris(Die2list)],
    ["Högsta pris", högstapris(Ben2list), högstapris(E852list), högstapris(Die2list)],
    ["Medelpris", Medelpris(Ben2list), Medelpris(E852list), Medelpris(Die2list)]
```

```
]

# print it as a table

for row in table:
    print("{:<15} {:<15} {:<15} {:<15}".format(*row))

##########################################
#Bensintid
def drivmedeltid(list1):
    counts={i: 0 for i in range(9, 19)}# bins 9-10, 10-11, ..., 18-19

    #Count values into bins
    for i in range(len(list1)):
        value=float(list1[i])
        for k in counts.keys():
            if k<= value <k+1:
                counts[k] +=1
                break

    #Normalize to frequency
    total=len(list1)
    freqs={k: round(counts[k]/total, 2) for k in counts}

    return freqs

bensin_freqs=drivmedeltid(Ben2list)
E85_freqs=drivmedeltid(E852list)
diesel_freqs=drivmedeltid(Die2list)

bins=range(9,20)#9-10, 10-11, 11-12
first_bensin={i: None for i in bins}
first_E85={i: None for i in bins}
first_diesel={i: None for i in bins}

# Loop over the fuel prices
for row in fuelprices[1:]:
    date = row[0]
    bensin_price=float(row[1])
    E85_price=float(row[2])
    diesel_price=float(row[3])

    for k in bins:
        if k<=bensin_price <k+1 and first_bensin[k] is None:
            first_bensin[k]=date
```

```
        if k<=E85_price <k+1 and first_E85[k] is None:
            first_E85[k]=date
        if k<=diesel_price <k+1 and first_diesel[k] is None:
            first_diesel[k]=date

# Print results

table=[["Prisinterfvall", "Bensin av mätperiod", "Bensin datum", "E85 av mätperiod", "E85

for k in bins:
    table.append([
      f"{k}.00-{k}.99",
      bensin_freqs.get(k,0), first_bensin[k],
      E85_freqs.get(k,0), first_E85[k],
      diesel_freqs.get(k,0), first_diesel[k]
    ])# om key finns, returnerar värdet, annars default 0


# print it as a table

for row in table:
    row_printable= [str(x) if x is not None else "--" for x in row]#Converting None to "--
    print("{:<15} {:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format(*row_printable))
```

```
                Bensin [kr/liter] E85 [kr/liter]  Diesel [kr/liter]
Lägsta pris     12.07             9.39            11.38
Högsta pris     17.84             15.49           19.32
Medelpris       14.53             11.42           14.54
Prisinterfvall  Bensin av mätperiod Bensin datum    E85 av mätperiod E85 datum       Diese
9.00-9.99       0.0               --              0.05             2015-01-01      0.0
10.00-10.99     0.0               --              0.34             2015-03-19      0.0
11.00-11.99     0.0               --              0.37             2015-11-09      0.03
12.00-12.99     0.13              2015-01-01      0.16             2017-02-13      0.14
13.00-13.99     0.23              2015-02-10      0.07             2019-11-21      0.26
14.00-14.99     0.29              2015-04-16      0.0              2021-09-28      0.19
15.00-15.99     0.22              2018-03-22      0.0              2021-10-06      0.15
16.00-16.99     0.13              2018-05-18      0.0              --              0.2
17.00-17.99     0.01              2019-05-17      0.0              --              0.02
18.00-18.99     0.0               --              0.0              --              0.01
19.00-19.99     0                 --              0                --              0
```