Aufgabe 4: Fahrradwerkstatt

Team-ID: 00464

Team: Enrique Lopez

Bearbeiter/-innen dieser Aufgabe:
Enrique Lopez

3. November 2022

Inhaltsverzeichnis

Teilaufgabe 1 -Programm

Lösungsidee

Der Grundaufbau des Programms ist für beide Methoden gleich. Es müssen zunächst die Auftragsdaten, dass heißt der Eingangszeitpunkt und die Bearbeitungsdauer, eingelesen werden.

Wir benötigen eine Schleife, bei welcher eine Iteration eine vergangene Minute bedeutet. In dieser Schleife soll dann entschieden werden, was Marc als nächstes macht:

1. Wenn er gerade an einem Job arbeitet, soll er daran weiterarbeiten und die restliche Dauer dieses Auftrags um eine weitere Minute verringern.

2. Wenn der aktuelle Job fertig ist so soll er sich einen neuen aus der Liste der zu diesem Zeitpunkt bereits eingegangenen Aufträgen aussuchen. Dieser Schritt ist je nach Methode unterschiedlich. Bei Methode 1 wird der Nächste in der Warteschlange gewählt, bei Methode 1 der kürzeste bereits Eingetroffene.

3. Wenn der aktuelle Job fertig ist und kein neuer eingetroffen ist, so soll Marc warten.

Dabei soll die vergangene Zeit seit dem ersten Kalendertag in einer Variable gespeichert werden.

Wenn um 17 Uhr Marcs Feierabend beginnt, kann dieser Zeit-Variable einfach die Minutenanzahl bis zum nächsten Morgen um 9 Uhr angehängt werden.

Sobald ein Auftrag erledigt ist, so wird die Wartezeit, also die vergangene Zeit seit dem Eingang, berechnet und gespeichert. Wenn alle Aufträge erledigt sind, so wird das Programm beendet und die Bilanz der Wartezeiten (durchschnittlich, größte) wird ausgegeben.

Umsetzung

Das Programm besteht aus einer Hauptfunktion worktime(), welche den Dateinamen der Textdatei entgegennimmt. In dieser Funktion werden zunächst der Eingangszeitpunkt und die Dauer jedes Auftrags jeweils in zwei Listen eingelesen. Daraufhin folgen zwei Funktionsdefinitionen, jeweils eine für die zwei Methoden zur Auswahl des nächsten zu bearbeitenden Auftrags.

Die erste Funktion earlybird() berechnet die durchschnittliche Wartezeit aller Aufträge bei Erledigung dieser in ihrer Eingangsreihenfolge. Sie nimmt die beiden Listen mit geschätzter Dauer und Eingangszeitpunkt als Argumente. Weiterhin definiert sie die Variable "time", die seit dem ersten Kalendertag um 0 Uhr vergangene Zeit, welche von Anfang an 540 ist, da bereits 540 Minuten vergangen sind, wenn Marc zu arbeiten beginnt.

Die Variable "currentjob" enthält die Bearbeitungsdauer in Minuten, welche beim aktuellen Auftrag verbleibt. Die Variable "feierabend" enthält den Zeitpunkt, bei den Marc das nächste mal Feierabend hat. Die Liste "wait_times" enthält am Ende die finale Wartezeit jedes Auftrags, sodass damit unter anderem der Durchschnitt berechnet werden kann.

Die Endlosschleife beginnt. Zunächst wird geprüft, ob Marcs aktuelles Projekt fertig ist bzw. ob er noch gar keins hat (currentjob = 0). Wenn dies der Fall ist, wird die Wartezeit des beendeten Auftrags berechnet und der Liste wait_times angehängt, sofern time != 0 ist, denn dann hatte Marc noch keinen Auftrag. Dann wird geprüft, ob alle Aufträge abgearbeitet wurden. Falls ja, werden die durchschnittliche, die größte und die kleinste Wartezeit aus der Liste ausgerechnet und durch String Formatting ausgegeben, woraufhin die Funktion beendet wird.

Gibt es noch einen zu bearbeitenden Auftrag in der Liste, welcher zu diesem Zeitpunkt bereits eingetroffen ist (time >=entry), so wird bei dieser Funktion der nächste in der Warteschlange gewählt und die Variable currentjob nimmt den Wert der geschätzten Dauer dieses Auftrags an. Die Auftragsdauer wird aus der Liste entfernt, wobei der Eingangszeitpunkt des Auftrags erst später bei der Berechnung der Wartezeit entfernt wird. Dieser Teil des Programms befindet sich in einer Endlosschleife, denn wenn noch kein neuer Auftrag eingetroffen ist, so wirkt die Schleife als "Warteschleife" und die Zeit wird so lange erhöht bis ein neuer Auftrag eingetroffen ist. Somit muss bei der Annahme eines neuen Auftrag durch "break" aus der Schleife herausgebrochen werden.

Zum Ende einer Iteration der Hauptschleife wird die Minutenzahl time um 1 erhöht und die Variable currentjob um 1 erniedrigt. Daraufhin wird zusätzlich geprüft, ob für Marc Feierabend beginnt, indem die time Variable mit der feierabend Variable verglichen wird. Stimmen die beiden überein, so ist es 17 Uhr und die Time Variable wird um 960 erhöht, sodass es wieder 9 Uhr ist und Marc weiterarbeiten kann. Zusätlich wird die feierabend Variable um 1440 erhöht, quasi auf den Zeitpunkt 17 Uhr des nächsten Tages.

Die zweite Funktion short_is_great() berechnet die durchschnittliche Wartezeit jedes Auftrags wenn Marc nach der Erledigung eines Auftrags immer den kürzesten zu diesem Zeitpunkt verfügbaren Auftrag wählt. Der Grundaufbau der Funktion ist ähnlich zu dem der ersten Funktion, jedoch sind einige Änderungen nötig. Zunächst müssen zu jeder Zeit die zu diesem Zeitpunkt verfügbaren Aufträge bekannt sein. Dies wird bewerkstelligt, indem sowohl in der Hauptschleife als auch in der Warteschleife jede Minute geprüft wird, ob der nächste Auftrag eingetroffen ist, und die Daten dieses gegebenenfalls in die neuen Listen waitlist_ready und entry_ready überträgen und in den ursprünglichen Listen gelöscht wird. So kann nun bei Neuauswahl eines Auftrags der kürzeste der zurzeit verfügbaren ermittelt und gewählt werden. Der Eingangszeitpunkt dieses Auftrags wird für die Berechnung der Wartezeit später in der Variable currententry gespeichert.

Zudem muss vor Abschluss des Programms nun überprüft werden, ob sowohl die Liste mit noch nicht eingetroffenen Aufträgen "entry" und die Liste der verfügbaren Aufträge "entry_ready" keine Objekte enthalten. Erst wenn dies der Fall ist hat Marc alle Aufträge erledigt und das Programm gibt die Bilanz der Wartezeiten aus. (Für Programm siehe Quellcode)

Beispiele

1. fahrradwerkstatt0.txt

```
In [74]: worktime('fahrradwerkstatt0.txt')
         Erste Methode: Erledigung in Reihenfolge

         Fertig nach 12.0 Monaten!
          Durchschnittliche Wartezeit: 2101 Minuten
          Längste Wartezeit: 8500 Minuten
          Kürzeste Wartezeit: 13 Minuten

          ------------------------------------------

         Zweite Methode: Kürzeste Aufträge haben Priorität

         Fertig nach 12.0 Monaten!
          Durchschnittliche Wartezeit: 2053 Minuten
          Längste Wartezeit: 8500 Minuten
          Kürzeste Wartezeit: 13 Minuten

          ------------------------------------------
```

- Die durchschnittliche Wartezeit der 2ten Methode ist ca. 97,7% der durchschnittlichen Wartezeit der ersten Methode

- Keine Unterschiede bei den anderen Kennzahlen

2. fahrradwerkstatt1.txt

```
In [107]: worktime("fahrradwerkstatt1.txt")

          Erste Methode: Erledigung in Reihenfolge

          Fertig nach 12.0 Monaten!
           Durchschnittliche Wartezeit: 487 Minuten
           Längste Wartezeit: 2638 Minuten
           Kürzeste Wartezeit: 13 Minuten

           ------------------------------------------

          Zweite Methode: Kürzeste Aufträge haben Priorität

          Fertig nach 12.0 Monaten!
           Durchschnittliche Wartezeit: 426 Minuten
           Längste Wartezeit: 2927 Minuten
           Kürzeste Wartezeit: 13 Minuten

           ------------------------------------------
```

–Die durchschnittliche Wartezeit der 2ten Methode ist ca. 87,4% der durchschnittlichen Wartezeit der 1ten Methode

–die längste Wartezeit der 2ten Methode ist ca. 111% der längsten Wartezeit der 1ten Methode

3. fahrradwerkstatt2.txt

```
In [75]: worktime('fahrradwerkstatt2.txt')

         Erste Methode: Erledigung in Reihenfolge

         Fertig nach 12.0 Monaten!
          Durchschnittliche Wartezeit: 1661 Minuten
          Längste Wartezeit: 9755 Minuten
          Kürzeste Wartezeit: 13 Minuten

          ------------------------------------------

         Zweite Methode: Kürzeste Aufträge haben Priorität

         Fertig nach 12.0 Monaten!
          Durchschnittliche Wartezeit: 1434 Minuten
          Längste Wartezeit: 10295 Minuten
          Kürzeste Wartezeit: 13 Minuten

          ------------------------------------------
```

–Die durchschnittliche Wartezeit der 2ten Methode ist ca. 94,7% der durchschnittlichen Wartezeit der ersten Methode

–die längste Wartezeit der 2ten Methode ist ca. 105,5% der längsten Wartezeit der 1ten Methode

4. fahrradwerkstatt3.txt

```
In [76]: worktime('fahrradwerkstatt3.txt')
```

```
Erste Methode: Erledigung in Reihenfolge

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 1102 Minuten
 Längste Wartezeit: 5319 Minuten
 Kürzeste Wartezeit: 13 Minuten

 ------------------------------------------

Zweite Methode: Kürzeste Aufträge haben Priorität

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 1076 Minuten
 Längste Wartezeit: 5831 Minuten
 Kürzeste Wartezeit: 13 Minuten

 ------------------------------------------
```

–Die durchschnittliche Wartezeit der 2ten Methode ist ca. 97,6% der durchschnittlichen Wartezeit der ersten Methode

–die längste Wartezeit der 2ten Methode ist ca. 109,6% der längsten Wartezeit der 1ten Methode

5. fahrradwerkstatt4.txt

```
In [77]: worktime('fahrradwerkstatt4.txt')
```

```
Erste Methode: Erledigung in Reihenfolge

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 3284 Minuten
 Längste Wartezeit: 11803 Minuten
 Kürzeste Wartezeit: 14 Minuten

 ------------------------------------------

Zweite Methode: Kürzeste Aufträge haben Priorität

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 2896 Minuten
 Längste Wartezeit: 18885 Minuten
 Kürzeste Wartezeit: 14 Minuten

 ------------------------------------------
```

–Die durchschnittliche Wartezeit der 2ten Methode ist ca. 88,1% der durchschnittlichen Wartezeit der ersten Methode

–die längste Wartezeit der 2ten Methode ist ca. 160% der längsten Wartezeit der 1ten Methode

6. grosser_unterschied.txt

```
In [103]: worktime("grosser_unterschied.txt")
          Erste Methode: Erledigung in Reihenfolge

          Fertig nach 1.4 Monaten!
           Durchschnittliche Wartezeit: 22540 Minuten
           Längste Wartezeit: 42290 Minuten
           Kürzeste Wartezeit: 550 Minuten

           ----------------------------------------

          Zweite Methode: Kürzeste Aufträge haben Priorität

          Fertig nach 1.4 Monaten!
           Durchschnittliche Wartezeit: 19574 Minuten
           Längste Wartezeit: 62820 Minuten
           Kürzeste Wartezeit: 10 Minuten

           ----------------------------------------
```

–Die durchschnittliche Wartezeit der 2ten Methode ist ca. 86,8% der durchschnittlichen Wartezeit der ersten Methode

–die längste Wartezeit der 2ten Methode ist ca. 160% der längsten Wartezeit der 1ten Methode

–Dieses Beispiel wurde erstellt, um das Hauptproblem der 2ten Methode darzustellen. Zum Zeitpunkt 0 gehen zwei Aufträge ein, einer mit einer geschätzten Dauer von 1000 Minuten und einer mit einer geschätzten Dauer von 10 Minuten. Ab dem Zeitpunkt 550 Minuten nach Start des Kalendertages trifft alle 10 Minuten ein Auftrag mit einer Dauer von 10 Minuten ein, und das insgesamt 2000 mal. Bei genauerer Analyse der Auftragsbearbeitung der 2ten Methode ist der 1000 Minuten Auftrag der erste der eintrifft, wird jedoch als letztes bearbeitet, da Marc immer den kürzeren verfügbaren 10 Minuten Auftrag wählt.

Teilaufgabe 2 –Unzufriedenheit

Das letzte Beispiel zeigt auf, warum vor allem Auftraggeber mit länger dauernden Aufträgen bei der 2ten Methode unzufrieden sein werden. Obwohl die durchschnittliche Wartezeit aller Aufträge bei Methode 2 in alle Beispielen geringer ist als bei Methode 1, so werden Auftraggeber mit langen Aufträgen benachteiligt und müssen im Durchschnitt länger warten, da kürzere Aufträge eine höhere Auswahlpriorität haben. Dies kann im Extremfall (letztes Beispiel) dazu führen, dass der erste eingegangene Auftrag aufgrund seiner Länge der letzte ist, der bearbeitet und fertiggestellt wird. Dies ist unfair dem Kunden gegenüber und auch ökonomisch nicht sehr schlau, da Marc vor allem die Zufriedenheit der Kunden mit längeren und damit teureren Aufträgen maximieren will, um diese wertvollen Kunden beizubehalten.

Teilaufgabe 3 – Beste Methode

Es gibt keine gerechtere und sinnvollere Methode als die Aufträge in Reihenfolge nach Eingangszeitpunkt zu bearbeiten. Marc wollte sein System ändern, da Kunden mit kurzen Aufträgen sich beschwerten und ihm sagten, dass ihre Aufträge nicht bearbeitet werden würden, da längere Aufträge die Werkstatt blockierten. Marc hätte erwidern sollen dass Kunden mit kurzen Aufträgen kein Vorrecht haben und jeder Kunde es verdient, seinen Auftrag nach Buchung sobald wie möglich erledigt zu bekommen. Das Sprichwort „Wer zu erst kommt malt zuerst“ trifft auf diese Situation zu und nur weil ein Auftrag kürzer ist als ein Auftrag, der schon vorher eingereicht wurde, sollte er keine Privilegien erhalten. Wenn Marc seine Werkstatt unbedingt attraktiv für Kunden mit kurzen Aufträgen machen will, dann darf er schlicht und ergreifend keine längeren Aufträge mehr annehmen, und muss sich vollständig auf schnelle Aufträge fokussieren. Jeder andere Algorithmus zur Auswahl des nächsten Auftrags wäre unfair gegenüber Kunden, die bereits früher gebucht haben. Das fokussieren auf bestimmte Metriken wie durchschnittliche Wartezeit spiegelt die Realität nur schlecht wieder und schadet letzten Endes der Kundenzufriedenheit mindestens einer Kundengruppe. So zum Beispiel kann Marc durch die 2te Methode die durchschnittliche Wartezeit nur dadurch erhöhen, dass er Kunden mit langen Aufträgen benachteiligt, wodurch sich die Wartezeit für sie deutlich erhöht. Daher bin ich der Ansicht dass jegliches neues System mit verschieden gewichteten Metriken zur Auswahl neuer Aufträge dem Kern des Auftrags von Marcs Werkstatt, nämlich das Fahrrad seiner Kunden so schnell wie möglich nach Buchung zu reparieren, schaden würde. Dies ist der Grund warum in jedem Unternehmen – von Restaurants bis Amazon – Aufträge und Bestellungen nach Eingangszsdatum bearbeitet werden.

Da ich für diese Aufgabe trotzdem etwas programmieren wollte, habe ich zwei zusätzliche Kennzahlen und zwei Methoden zur Auswahl des nächsten Auftrags hinzugefügt:

Wartezeit-Unterschied: Unterschied zwischen längster und kürzester Wartezeit. Sollte für Fairness und Kundenzufriedenheit minimal gehalten werden.

Median: Mittelwert aller Wartezeiten. Gleichzeitig ein Messwert für Unterschied zwischen längster und kürzester Wartezeit, beinhaltet jedoch auch die Minimierung der Wartezeit für alle Kunden.

Zufällige Selektion: Marc wählt einen zufälligen Auftrag aus den verfügbaren.

Letzter Auftrag: Marc wählt immer den zuletzt eingetroffenen Auftrag.

Siehe nächste Seite für Beispiele

Die Performance der dritten, zufälligen Methode ist zufällig und kann daher schlecht verglichen werden, jedoch lässt sich nach einigen Tests und Wiederholungen der Methode feststellen, dass sie bis auf einige Ausnahmen meistens schlechtere Ergebnisse liefert als die Methoden 1 und 2.

Auch die vierte Methode liefert enttäuschende Ergebnisse. Die Auswahl des letzten eingetroffenen Auftrags sorgt für extrem lange Wartezeiten bei einigen Kunden, wodurch auch die Durchschnittliche Wartezeit erhöht ist.

Während der Wartezeit-Unterschied akkurat die Probleme der 2ten und 4ten Methode darstellen kann, stellt sich der Median als eher schlechte Kennzahl zur Leistungsmessung heraus. Vor allem bei Methode 4 würde der Median suggerieren, dass die Methode gut funktioniert, während die durchschnittliche Wartezeit sogar bei Methode 1 niedriger ist.

In [6]:
```
worktime("fahrradwerkstatt0.txt")
```

```
-------------------------------------------

Erste Methode: Erledigung in Reihenfolge

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 2101 Minuten
 Längste Wartezeit: 8500 Minuten
 Wartezeit Unterschied: 8487 Minuten
 Median: 1528 Minuten

-------------------------------------------

Zweite Methode: Kürzeste Aufträge haben Priorität

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 2053 Minuten
 Längste Wartezeit: 8500 Minuten
 Wartezeit-Unterschied: 8487 Minuten
 Median: 1512 Minuten

-------------------------------------------

Dritte Methode: Zufällige Auswahl aus verfügbaren Aufträgen

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 2139 Minuten
 Längste Wartezeit: 8500 Minuten
 Wartezeit-Unterschied: 8487 Minuten
 Median: 1522 Minuten

-------------------------------------------

Vierte Methode: Letzter eingetroffener Auftrag wird bearbeitet
Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 2146 Minuten
 Längste Wartezeit: 11528 Minuten
 Wartezeit-Unterschied: 11515 Minuten
 Median: 1522 Minuten
```

:
```
worktime("fahrradwerkstatt1.txt")
```

```
-------------------------------------------

Erste Methode: Erledigung in Reihenfolge

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 487 Minuten
 Längste Wartezeit: 2638 Minuten
 Wartezeit Unterschied: 2625 Minuten
 Median: 239 Minuten

-------------------------------------------

Zweite Methode: Kürzeste Aufträge haben Priorität

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 426 Minuten
 Längste Wartezeit: 2927 Minuten
 Wartezeit-Unterschied: 2914 Minuten
 Median: 209 Minuten

-------------------------------------------

Dritte Methode: Zufällige Auswahl aus verfügbaren Aufträgen

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 522 Minuten
 Längste Wartezeit: 5126 Minuten
 Wartezeit-Unterschied: 5113 Minuten
 Median: 234 Minuten

-------------------------------------------

Vierte Methode: Letzter eingetroffener Auftrag wird bearbeitet
Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 545 Minuten
 Längste Wartezeit: 5807 Minuten
 Wartezeit-Unterschied: 5794 Minuten
 Median: 236 Minuten
```

```
worktime("fahrradwerkstatt3.txt")
```

```
-------------------------------------------

Erste Methode: Erledigung in Reihenfolge

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 1102 Minuten
 Längste Wartezeit: 5319 Minuten
 Wartezeit Unterschied: 5306 Minuten
 Median: 1308 Minuten

-------------------------------------------

Zweite Methode: Kürzeste Aufträge haben Priorität

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 1076 Minuten
 Längste Wartezeit: 5831 Minuten
 Wartezeit-Unterschied: 5818 Minuten
 Median: 1301 Minuten

-------------------------------------------

Dritte Methode: Zufällige Auswahl aus verfügbaren Aufträgen

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 1107 Minuten
 Längste Wartezeit: 5831 Minuten
 Wartezeit-Unterschied: 5818 Minuten
 Median: 1308 Minuten

-------------------------------------------

Vierte Methode: Letzter eingetroffener Auftrag wird bearbeitet
Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 1124 Minuten
 Längste Wartezeit: 7458 Minuten
 Wartezeit-Unterschied: 7445 Minuten
 Median: 1308 Minuten
```

[3]:
```
worktime("fahrradwerkstatt4.txt")
```

```
-------------------------------------------

Erste Methode: Erledigung in Reihenfolge

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 3284 Minuten
 Längste Wartezeit: 11803 Minuten
 Wartezeit Unterschied: 11789 Minuten
 Median: 3226 Minuten

-------------------------------------------

Zweite Methode: Kürzeste Aufträge haben Priorität

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 2896 Minuten
 Längste Wartezeit: 18885 Minuten
 Wartezeit-Unterschied: 18871 Minuten
 Median: 2990 Minuten

-------------------------------------------

Dritte Methode: Zufällige Auswahl aus verfügbaren Aufträgen

Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 3160 Minuten
 Längste Wartezeit: 14404 Minuten
 Wartezeit-Unterschied: 14390 Minuten
 Median: 3181 Minuten

-------------------------------------------

Vierte Methode: Letzter eingetroffener Auftrag wird bearbeitet
Fertig nach 12.0 Monaten!
 Durchschnittliche Wartezeit: 3057 Minuten
 Längste Wartezeit: 18885 Minuten
 Wartezeit-Unterschied: 18871 Minuten
 Median: 2990 Minuten
```

Quellcode

November 4, 2022

[ ]:
```
def worktime(datei):
    #Erspart einige Dinge wie Durchschnitt von Werten in Array
    import numpy as np

    #Für Methode 3
    import random

    #Listen für Eingangszeit und Dauer
    entry = []
    waitlist = []
    with open(datei) as f:
        line = f.readline()
        while line:
            entry.append(int(line.split()[0]))
            waitlist.append(int(line.split()[1]))
            line = f.readline()

    ##########################################################################

    #ERLEDIGUNG IN REIHENFOLGE
    def earlybird(entry,waitlist):


        #Variable der insgesamt vergangenen Zeit (beginnt um 9 Uhr)
        time = 540

        #Variable für verbleibende Zeit am aktuellen Projekt
        currentjob = 0

        #Zeitpunkt an dem Marc Feierabend hat
        feierabend = 1020

        #Wartedauern aller Aufträge
        wait_times = []

        #Äußerer Produktiver Loop
        while True:
```

```
            #System für Auswahl: Nächster in der Schlange
            if currentjob <= 0:

                #Dauer des letzten Auftrags wird berechnet
                if time != 540:
                    wait_times.append(time-currententry)


                #Alle Aufträge fertig
                if len(entry) == 0:
                    wait_times = np.array(wait_times)
                    return print('Fertig nach {} Monaten! \n Durchschnittliche␣
↪Wartezeit: {} Minuten \n Längste Wartezeit: {} Minuten \n Wartezeit␣
↪Unterschied: {} Minuten \n Median: {} Minuten'.format(round(time * 0.
↪000022816,1),int(round(sum(wait_times) /␣
↪len(wait_times),0)),max(wait_times),max(wait_times)-min(wait_times),int(round(np.
↪median(wait_times),0))))

                #Warteloop auf nächsten Auftrag
                while True:
                    if entry[0] <= time:
                        currentjob = waitlist[0]
                        currententry = entry[0]
                        del waitlist[0]
                        del entry[0]
                        break
                    else:
                        time +=1

            time += 1
            currentjob -= 1

            #Wenn Feierabend ist
            if time == feierabend:
                time += 960
                feierabend += 1440
    ##########################################################################

    def short_is_great(entry,waitlist):
        #Variable der insgesamt vergangenen Zeit (beginnt um 9 Uhr)
        time = 540

        #Variable für verbleibende Zeit am aktuellen Projekt
        currentjob = 0

        #Zeitpunkt an dem Marc Feierabend hat
```

```
        feierabend = 1020

        #Wartedauern aller Aufträge
        wait_times = []

        #Liste der verfügbaren Aufträge
        waitlist_ready = []
        entry_ready = []

        #Äußerer Produktiver Loop
        while True:

            if len(entry) != 0:
                if time >= entry[0]:
                    waitlist_ready.append(waitlist[0])
                    entry_ready.append(entry[0])
                    del waitlist[0]
                    del entry[0]


            #System für Auswahl: Nächster in der Schlange
            if currentjob <= 0:

                #Dauer des letzten Auftrags wird berechnet
                if time != 540:
                    wait_times.append(time-currententry)



                #Alle Aufträge fertig
                if len(entry_ready) == 0 and len(entry) == 0:
                    wait_times = np.array(wait_times)
                    return print('Fertig nach {} Monaten! \n Durchschnittliche␣
↪Wartezeit: {} Minuten \n Längste Wartezeit: {} Minuten \n␣
↪Wartezeit-Unterschied: {} Minuten \n Median: {} Minuten'.format(round(time *␣
↪0.000022816,1),int(round(sum(wait_times) /␣
↪len(wait_times),0)),max(wait_times),max(wait_times)-min(wait_times),int(round(np.
↪median(wait_times),0))))



                #Warteloop auf nächsten Auftrag
                while True:
                    #Index of shortest job
                    if len(waitlist_ready) > 0:
                        min_index = waitlist_ready.index(min(waitlist_ready))
```

```
                    currentjob = min(waitlist_ready)
                    currententry = entry_ready[min_index]
                    del waitlist_ready[min_index]
                    del entry_ready[min_index]
                    break

                else:
                    time +=1
                    if len(entry) != 0:
                        if time >= entry[0]:
                            waitlist_ready.append(waitlist[0])
                            entry_ready.append(entry[0])
                            del waitlist[0]
                            del entry[0]

        time += 1
        currentjob -= 1

        #Wenn Feierabend ist
        if time == feierabend:
            time += 960
            feierabend += 1440



def zufällig(entry,waitlist):

    #Variable der insgesamt vergangenen Zeit (beginnt um 9 Uhr)
    time = 540

    #Variable für verbleibende Zeit am aktuellen Projekt
    currentjob = 0

    #Zeitpunkt an dem Marc Feierabend hat
    feierabend = 1020

    #Wartedauern aller Aufträge
    wait_times = []

    #Liste der verfügbaren Aufträge
    waitlist_ready = []
```

```
        entry_ready = []

        #Äußerer Produktiver Loop
        while True:

            if len(entry) != 0:
                if time >= entry[0]:
                    waitlist_ready.append(waitlist[0])
                    entry_ready.append(entry[0])
                    del waitlist[0]
                    del entry[0]


            #System für Auswahl: Nächster in der Schlange
            if currentjob <= 0:

                #Dauer des letzten Auftrags wird berechnet
                if time != 540:
                    wait_times.append(time-currententry)



                #Alle Aufträge fertig
                if len(entry_ready) == 0 and len(entry) == 0:
                    wait_times = np.array(wait_times)
                    return print('Fertig nach {} Monaten! \n Durchschnittliche␣
↪Wartezeit: {} Minuten \n Längste Wartezeit: {} Minuten \n␣
↪Wartezeit-Unterschied: {} Minuten \n Median: {} Minuten'.format(round(time *␣
↪0.000022816,1),int(round(sum(wait_times) /␣
↪len(wait_times),0)),max(wait_times),max(wait_times)-min(wait_times),int(round(np.
↪median(wait_times),0))))



                #Warteloop auf nächsten Auftrag
                while True:
                    #Index of random Object
                    if len(waitlist_ready) > 0:
                        random_index = random.randrange(len(waitlist_ready))







                        currentjob = waitlist_ready[random_index]
                        currententry = entry_ready[random_index]
```

```
                    del waitlist_ready[random_index]
                    del entry_ready[random_index]
                    break

                else:
                    time +=1
                    if len(entry) != 0:
                        if time >= entry[0]:
                            waitlist_ready.append(waitlist[0])
                            entry_ready.append(entry[0])
                            del waitlist[0]
                            del entry[0]

        time += 1
        currentjob -= 1

        #Wenn Feierabend ist
        if time == feierabend:
            time += 960
            feierabend += 1440


def letzter_auftrag(entry,waitlist):

    #Variable der insgesamt vergangenen Zeit (beginnt um 9 Uhr)
    time = 540

    #Variable für verbleibende Zeit am aktuellen Projekt
    currentjob = 0

    #Zeitpunkt an dem Marc Feierabend hat
    feierabend = 1020

    #Wartedauern aller Aufträge
    wait_times = []

    #Liste der verfügbaren Aufträge
    waitlist_ready = []
    entry_ready = []

    #Äußerer Produktiver Loop
    while True:

        if len(entry) != 0:
            if time >= entry[0]:
                waitlist_ready.append(waitlist[0])
                entry_ready.append(entry[0])
```

```
                del waitlist[0]
                del entry[0]


        #System für Auswahl: Nächster in der Schlange
        if currentjob <= 0:

            #Dauer des letzten Auftrags wird berechnet
            if time != 540:
                wait_times.append(time-currententry)



            #Alle Aufträge fertig
            if len(entry_ready) == 0 and len(entry) == 0:
                wait_times = np.array(wait_times)
                return print('Fertig nach {} Monaten! \n Durchschnittliche␣
↪Wartezeit: {} Minuten \n Längste Wartezeit: {} Minuten \n␣
↪Wartezeit-Unterschied: {} Minuten \n Median: {} Minuten'.format(round(time *␣
↪0.000022816,1),int(round(sum(wait_times) /␣
↪len(wait_times),0)),max(wait_times),max(wait_times)-min(wait_times),int(round(np.
↪median(wait_times),0))))



            #Warteloop auf nächsten Auftrag
            while True:
                #Index of random Object
                if len(waitlist_ready) > 0:

                    currentjob = waitlist_ready[-1]
                    currententry = entry_ready[-1]
                    del waitlist_ready[-1]
                    del entry_ready[-1]
                    break

                else:
                    time +=1
                    if len(entry) != 0:
                        if time >= entry[0]:
                            waitlist_ready.append(waitlist[0])
                            entry_ready.append(entry[0])
                            del waitlist[0]
                            del entry[0]

        time += 1
        currentjob -= 1
```

```
            #Wenn Feierabend ist
            if time == feierabend:
                time += 960
                feierabend += 1440


    ##########################################################################
    print("\n ------------------------------------------ \n")
    print(' Erste Methode: Erledigung in Reihenfolge \n')
    earlybird(entry.copy(),waitlist.copy())

    print("\n ------------------------------------------ \n")

    print('Zweite Methode: Kürzeste Aufträge haben Priorität \n')
    short_is_great(entry.copy(),waitlist.copy())

    print("\n ------------------------------------------ \n")

    print('Dritte Methode: Zufällige Auswahl aus verfügbaren Aufträgen \n')
    zufällig(entry.copy(),waitlist.copy())

    print("\n ------------------------------------------ \n")

    print('Vierte Methode: Letzter eingetroffener Auftrag wird bearbeitet')
    letzter_auftrag(entry.copy(),waitlist.copy())

worktime(input('Dateiname der Textdatei mit den Auftragsdaten: '))
```

[ ]: