

Institutt for datateknikk og informasjonsvitenskap

## Eksamensoppgave i TDT4110 Informasjonsteknologi – grunnkurs, med Python – LØSNINGSFORSLAG

### Løsningsforslag for følgende oppgaver:

- Oppgave 1: Flervalgsoppgave (25%)
- Oppgave 2: Kodeforståelse (15%)
- Oppgave 3: Programmering reisetid (20%)
- Oppgave 4: Programmering sensur (40%)
- Svarark for hurtigsensur

### *Oppgave 1: Flervalgsoppgave (25%)*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
b	b	c	d	b	a	c	a	b	c	b	d	a	c	b	a	a	d	a	a

### *Oppgave 2 Kodeforståelse (15%)*

#### **Oppgave 2a (5%)**

Hva blir skrevet ut til skjerm når du kjører koden som vist under? (3 %)

```
JULENISSEN
```

 Forklar med en setning hva funksjonen **mystery** gjør (2 %)

Funksjonen plukker ut annenhver bokstav fra de to listene og starter med liste B.

#### **Oppgave 2b (5%)**

Hva blir skrevet ut til skjerm når du kjører koden som vist under? (3 %)

```
64
```

 Forklar med en setning hva funksjonen **compute** gjør (2 %)

 Rekursiv funksjon som multipliserer et tall med 2 ganger tallet så lenge tallet er mindre enn 10,  $1*2*4*8 = 64$ .

**Oppgave 2c (5%)**

Hva blir skrevet ut til skjerm når du kjører koden som vist under? (3 %)

[8, 7, 6, 5, 3, 2, 1]

Forklar med en setning hva funksjonen **a** gjør (2 %)

Sorterer en liste med tall i synkende rekkefølge.

**Oppgave 3 Programmering reisetid (20%)****Oppgave 3a (5%)**Lag funksjonen **readTime** ...

```
def readTime():
    hour = -1
    while hour < 0 or hour > 23:
        hour = int(input("Enter hour: "))
        if hour < 0 or hour > 23:
            print("- ERROR: Hour must be between 0 and 23!")
    minute = -1
    while minute < 0 or minute > 59:
        minute = int(input("Enter minute: "))
        if minute < 0 or minute > 59:
            print("- ERROR: Minute must be between 0 and 59!")
    sec = -1
    while sec < 0 or sec > 59:
        sec = int(input("Enter second: "))
        if sec < 0 or sec > 59:
            print("- ERROR: Second must be between 0 and 59!")
    return [hour, minute, sec]
```

Alternativt:

```
def readValidInt(unit, low, high):
    number = int(input('Enter ' + unit.lower() + ': '))
    while number < low or number > high:
        print('- ERROR:', unit, 'must be between', low, 'and', high, '!')
        number = int(input('Enter ' + unit.lower() + ': '))
    return number

def readTime():
    hour = readValidInt('Hour', 0, 23)
    minute = readValidInt('Minute', 0, 59)
    sec = readValidInt('Second', 0, 59)
    return [hour, minute, sec]
```

**Oppgave 3b (5%)**Lag funksjonen **convertTime** ...

```
def convertTime(time, mode):
    if mode == 'time':
        hour = time // 3600
        time = time - (hour * 3600)
        minute = time // 60
        time = time - (minute * 60)
        sec = time
        return [hour, minute, sec]
    elif mode == 'sec':
        return (time[0] * 3600 + time[1] * 60 + time[2])
```

**Oppgave 3c (5%)**Lag funksjonen **travelTime** ...

```
def travelTime():
    print("Give departure time in hour, minute and second:")
    starttime=readTime()
    stoptime=[0,0,0]
    while (convertTime(starttime,'sec')>convertTime(stoptime,'sec')):
        print("Give arrival time in hour, minute and second:")
        stoptime=readTime()
        if (convertTime(starttime,'sec')>convertTime(stoptime,'sec')):
            print("- ERROR: Arrival time must be later than Departure time")
    traveltime = convertTime(stoptime,'sec')-convertTime(starttime,'sec')
    travelTab = convertTime(traveltime,'time')
    print('Travelttime:',travelTab[0],'hours',' ',travelTab[1],'min',' ',travelTab[2],'sec')
```

**Oppgave 3d (5%)**Lag funksjonen **analyzeBusRoutes** ...

```
def busTime(BusRoute):
    return convertTime(BusRoute[3:5]+[0],'sec')-convertTime(BusRoute[1:3]+[0],'sec')

def analyzeBusRoutes(BusTables):
    slowestTimeSec = fastestTimeSec = busTime(BusTables[0])
    slowestBus = fastestBus = BusTables[0][0]
    for BusRoute in BusTables:
        if busTime(BusRoute) > slowestTimeSec:
            slowestTimeSec = busTime(BusRoute)
            slowestBus = BusRoute[0]
        elif busTime(BusRoute) < fastestTimeSec:
            fastestTimeSec = busTime(BusRoute)
            fastestBus = BusRoute[0]
    slowestTime = convertTime(slowestTimeSec,'time')
    fastestTime = convertTime(fastestTimeSec,'time')
    print("The slowest bus route is bus nr.",slowestBus,'and it takes',
          slowestTime[0],'hour, ',slowestTime[1],'min.')
    print("The fastest bus route is bus nr.",fastestBus,'and it takes',
          fastestTime[0],'hour, ',fastestTime[1],'min.')
```

**Oppgave 4 Programmering Sensur (40%)****Oppgave 4 a) (5%)**

Lag starten på hovedprogrammet (ikke funksjon)...

```
NTNU_scores = (89,77,65,53,41,0)
NTNU_letters = ('A','B','C','D','E','F')
TASKS = ('1','2a','2b','2c','3a','3b','3c','3d','4a','4b','4c','4d','4e','4f','4g','4h')
WEIGHTS = tuple([25]+(15*[5]))
```

**Oppgave 4 b) (5%)**Lag funksjonen **makeArray** ...

```
def makeArray(Numbers,Texts):
    ReturnList=[]
    for i in range(len(Numbers)):
        ReturnList+= [[Numbers[i],Texts[i]]]
    return ReturnList
```

**Oppgave 4 c) (5%)**Lag funksjonen `computeScore` ...

```
def computeScore(Points, WEIGHTS):
    score=0
    for i in range(len(Points)):
        score+=Points[i]*WEIGHTS[i]
    return score/10
```

**Oppgave 4 d) (5%)**Skriv en funksjon `score2Letter` ...

```
def score2Letter(scoreSum, limitLetters):
    for item in limitLetters:
        if scoreSum>item[0]:
            return item[1]
```

**Oppgave 4 e) (5%)**Skriv en funksjon `addCandidate` ...

```
def addCandidate(candidateNumber, Scores, WEIGHTS):
    scoreSum=computeScore(Scores, WEIGHTS)
    try:
        f = open('eksamen.txt', 'a')
        s = str(candidateNumber)
        for number in Scores:
            s+="\t"+str(number)
        s+="\t"+str(round(scoreSum, 1))+"\n"
        f.write(s)
        f.close()
    except Exception as errorMessage:
        print(errorMessage)
```

**Oppgave 4 f) (5%)**Skriv en funksjon `readResultFile` ...

```
# Convert a table of strings to table of int
def nummarizeTable(Table):
    for i in range(len(Table)):
        if (i<len(Table)-1):
            Table[i]=int(Table[i])
        else:
            Table[i]=float(Table[i])
    return Table

def readResultFile(filename):
    f = open(filename, 'r')
    results = []
    for line in f:
        line=line.strip()
        listline = line.split('\t')
        listline=nummarizeTable(listline)
        results+=listline
    f.close()
    return results
```

**Oppgave 4 g) (5%)**Skriv en funksjon **checkResultOK** ...

```
def checkResultOK(filename, WEIGHTS):
    allOK = True
    results = readResultFile(filename)
    count={}
    for line in results:
        if(max(line[1:-1])>10 or min(line[1:-1])<0):
            print("ERROR: Candidate",line[0],"scores are not between 0-10!")
            allOK = False
        if line[-1]!=computeScore(line[1:-1],WEIGHTS):
            print("ERROR: Candidate",line[0],"has wrong total score!")
            allOK = False
        count[line[0]]=count.get(line[0],0) +1
    for key in count:
        if count[key]>1:
            print("ERROR: Candidate",key,"appears more than once!")
            allOK = False
    return allOK
```

**Oppgave 4 h) (5%)**Skriv en funksjon **listAll** ...

```
def listAll(filename,limitLetters):
    results = readResultFile(filename)
    results.sort() # Sorterer etter kandidatnr
    count = 0
    for line in results:
        grade = score2Letter(line[-1], limitLetters)
        print(str(line[0]),str(round(line[-1],1)).rjust(5),grade)
        count+=1
    return count
```

***Transparent for flervalgsoppgave: Viser riktige svar (dekker over feil svar)***

Kandidatnummer: \_\_\_\_\_ Program: \_\_\_\_\_

Fagkode: \_\_\_\_\_ Dato: \_\_\_\_\_

Antall sider: \_\_\_\_\_ Side: \_\_\_\_\_

<b><i>Oppgavenr</i></b>	<b><i>A</i></b>	<b><i>B</i></b>	<b><i>C</i></b>	<b><i>D</i></b>
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				

**Transparent for flervalgsoppgave: Viser feil svar (dekker over riktig svar)**

Kandidatnummer: \_\_\_\_\_ Program: \_\_\_\_\_

Fagkode: \_\_\_\_\_ Dato: \_\_\_\_\_

Antall sider: \_\_\_\_\_ Side: \_\_\_\_\_

<i>Oppgavenr</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
1.1		■		
1.2	■	■	■	■
1.3			■	
1.4	■	■	■	■
1.5		■		
1.6	■	■	■	■
1.7			■	
1.8	■	■	■	■
1.9		■		
1.10	■	■	■	■
1.11		■		
1.12	■	■	■	■
1.13	■			
1.14	■	■	■	■
1.15		■		
1.16	■	■	■	■
1.17	■			
1.18	■	■	■	■
1.19	■			
1.20	■	■	■	■