Løsningsforslag TDT4110 Informasjonsteknologi, grunnkurs 1.desember 2020

Oppgave 1.1 (5%)
Forklar kort, komplett og konsist sammenhengen mellom et høynivå programmeringsspråk som f.eks. Python og maskinkode.

Løsning:
Et **høynivå programmeringsspråk** som Python er laget slik at det skal være l**ett å lære seg og forstå for mennesker**, samt at det skal være **effektivt å programmere**. Det vil si i praksis at man kan **utføre mange instruksjoner i en datamaskin med relativt få linjer kode**. **Maskinkode** består av instruksjoner som kan **forstås og kjøres direkte av en prosessor**. Oppbygningen av et maskinkodespråk er **direkte knyttet opp til hvordan en prosessor er oppbygd** og hvilke instruksjoner den tilbyr. For mennesker er det mer krevende å skrive maskinkode ettersom det krever en forståelse om hvordan prosessoren fungerer i detalj. I tillegg må man skrive relativt skrive mye mer kode for å få utført operasjoner i en datamaskin sammenliknet med et høynivå programmeringsspråk. For å kunne kjøre kode skrevet i et høynivåprogrammeringsspråk, må denne koden først **oversettes** ved hjelp av en **tolker** eller **kompilator** som gjør den om til maskinkode.

Oppgave 1.2 (5%)
Forklar kort, komplett og konsist hvilke begrensninger antall bit et tall er representert med medfører (forklar med bruk av eksempler):

Løsning:
Det er t**o hoved-begrensninger** relatert til antall bit et tall er representer med. Den ene **begrensningen er relatert til hvor store tall man kan regne med**. Hvis man f.eks. har en variabel for heltall som er representert med 8 bit, kan man ikke regne med større verdier enn 255. Hvis man da skal utføre addisjonen 250 + 6 der variabelen man lagerer resultatet er representert med 8 bit, får man ikke tallet 256, men feil tall fordi man får overflyt (man har ikke nok bits til å representere tallet 256).
Den **andre begrensingen går på nøyaktighet** når man f.eks. gjøre beregninger med **flyttall**. Manglende antall bit til å representere desimalene kan føre til **avrundingsfeil** når man f.eks. utfører multiplikasjon, subtraksjon eller addisjon.

Oppgave 1.3 (5%)
Sammenlign kort, komplett og konsist de fire lokalnettverktopologiene buss, ring, mesh og stjerne:

Løsning:
Buss-nettverk består av **en kabel som alle datamaskinene er koplet til**, der alle datamaskiner som er koplet til nettverket kan sende signaler og motta signaler når som helst. **Utfordringen** med buss er å **koordinere hvem som skal sende et signal**, ettersom bare e**n kan gjøre det av gangen**.
I et **Ring-nettverk** er alle datamaskinene koplet sammen i en **lukket ring (loop)**, der første datamaskin er koplet til andre, andre til tredje osv. Kommunikasjon foregår ved at kommunikasjonen går i ring (**signalene blir sendt ut**). En mulig ulempe med et slikt nettverk er at **alt stopper opp hvis en av datamaskinene henger seg eller er avslått**. Ofte bruker man derfor eget utstyr som man kopler datamaskinen til isteden for direkte kopling mellom datamaskiner.
I et **Mesh-nettverk** er det **direkte koplinger mellom hver par av datamaskiner**. Den store **ulempen med dette er at man antall koplinger** mellom datamaskiner ikke skalerer. **Fordelen er stabilitet** og at hele nettverket ikke faller sammen hvis deler av nettverket faller ut.
I et **stjernenettverk** er alle **datamaskinene koplet sammen til et sentralt punkt** som kalles en hub. En slik organisering er ganske **enkelt å sette opp**, men en ulempe er at **huben er single-point-of-failure** (hvis den faller ned, stopper alt opp).

Oppgave 1.4 (5%)
Forklar kort, komplett og konsist hvilke mekanismer i TCP som sikrer overføring av data over internett:

Løsning:
TCP tilbyr følgende: Koplingsorienterte tjenester som gir **ende-til-ende tjenester** (**sikrer at data kommer fra avsender til mottaker**) slik at det virker som at man har en direkte fysisk kopling mellom de to. TCP tilbyr også dataoverføring som er **garantert**, slik at data som blir sendt er **komplett og i riktig rekkefølge**. TCP gjør det mulig å overføre data i **begge retninger** og lar applikasjoner enten å motta eller sende data når som helst. TCP gjør det også mulig å **strømme data** som f.eks. lyd og video via et strømme-interface. TCP tilbyr to applikasjoner en **sikker oppstart av kommunikasjon og avslutting eller stenging av en forbindelse**. For å få til dette inneholder TCP **deteksjon av feil i data** som er overført og datapakker som ikke har kommet fram og **ny sending av datapakker**.

Oppgave 1.5 (5%)
Sammenlign kort, komplett og konsist sikkerhetsangrepene wiretapping, replay, buffer overflow og portskanning:

Løsning:
Wiretapping og replay har litt samme karakteristikker ettersom man **benytter seg av informasjon sendt av andre brukere**. Den største forskjellen her er at for **wiretapping er det ikke spesifisert hva de kopierte datapakkene skal brukes til**, men for r**eplay bruker man bevisst innloggingsinformasjon** til å få tilgang som man ikke skal ha. **Replay, buffer overflow og portskanning har samme formål** – i **å bryte seg inn i et system** – men med ulike teknikker. Både**buffer overflow og portskanning benytter seg av svakheter med sikkerheten på en server** – der **overflow benytter seg av usikker koding** (ikke beskyttelse mot mer input enn systemet tåler), og for **portskanning er det mangel på rutiner og oppfølging** som gjør at porter står åpne slik at de kan utnyttes av personer som ikke skal ha tilgang.

Oppgave 2.1 (5%)
Lag en funksjon **car_registration** som genererer og returnerer et tilfeldig bilnummer.

Løsningsforslag:
```
import random

def car_registration():
    boks = 'abcdefghijklmnopqrstuvwxyz'
    nummer = ''
    for i in [0,1]:
        nummer += random.choice(boks).upper()
    for i in range(5):
        siffer = random.randint(0,9)
        while i == 0 and siffer == 0:
           siffer = random.randint(0,9)
        nummer += str(siffer)
    return nummer
```

Oppgave 2.2 (5%)
Lag en funksjon **prepare** som tar inn et binært tall og forbereder det til å bli konvertert til oktalt.

Løsningsforslag:
```
def prepare(tekst):
    for tegn in tekst:
        if tegn not in '01':
            return 'Det er ikke et binært tall.'
    if int(len(tekst)) % 3 == 0:
        return tekst
    elif int(len(tekst)) % 3 == 1:
        return '00'+tekst
    else:
        return '0'+tekst
```

Oppgave 2.3 (5%)
Lag funksjonen **convert** som tar inn en tekststreng, konverterer tegnene til sine tilsvarende tallverdier *(vi har gjerne kalt dette å finne ASCII-verdien, men det virker for alle tegn i UNICODE)*, summerer disse sammen og så returnerer UNICODE-tegnet som tilsvarer tallet du er kommet fram til.

Løsningsforslag:
```
def convert(tekst):
    if len(tekst) > 5:
        tekst = tekst[-5:]
    summ = 0
    for tegn in tekst:
        summ += ord(tegn)
    return chr(summ)
```

Oppgave 2.4 (5%)
Skriv funksjonen **add_matrix** som legger sammen 2 matriser/2D-lister og returnerer resultatet.

Løsningsforslag:
```
# Med numpy
import numpy as np
def add_matrix(c,d):
    matr1 = np.array(c)
    matr2 = np.array(d)
    if matr1.shape == matr2.shape:
        return matr1+matr2
    else:
        print('Kan ikke summere.')

# Uten numpy
def add_matrix2(c,d):
    ny = []
    if len(c) == len(d) and len(c[0]) == len(d[0]):
        for i in range(len(c)):
            liste = []
            for j in range(len(c[0])):
                nytt = c[i][j] + d[i][j]
                liste.append(nytt)
            ny.append(liste)
    return ny
```

Oppgave 2.5 (5%)
Du har data lagret som en 2D-liste der hvert element er [band,tittel]
Disse dataene skal lagres til disk som en tekstfil. Skriv funksjonen **list_to_disk()** som tar 2D-listen og et filnavn som input og lagrer dataene på den oppgitte filen på følgende format:
band;låttittel

Løsningsforslag:
```
def list_to_disc(liste,filename):
    f = open(filename,'w')
    for element in liste:
        streng = ';'.join(element) + '\n'
        f.write(streng)
    f.close()
```

Oppgave 3.1 (5%)
Skriv funksjonen *calculate_percentage()*. Funksjonens parameter er et *tuppel*, som inneholder antallet stemmer demokratene har fått, etterfulgt av antallet stemmer republikanerne har fått.

Løsningsforslag:
```
def calculate_percentage(votes):
    dem = int(votes[0])*100/(int(votes[0]) + int(votes[1]))
    rep = 100 - dem
    return round(dem,2),round(rep,2)
```

Oppgave 3.2 (5%)
Lag funksjonen return_total_votes(). Den skal summere antallet stemmer demokratene fikk (det første tallet i hvert tuppel) og republikanerne (det siste tallet i hvert tuppel).

Kommentar: Eksemplet i oppgaven viste bruk av to parametere og ikke et tuppel. Begge løsninger må godtas.

Løsningsforslag:
```
def return_total_votes(state_votes):
    dem, rep = 0,0
    for i in state_votes:
        dem += int(i[0])
        rep += int(i[1])
    return (dem,rep)
```

Oppgave 3.3 (5%)
Skriv funksjonen update_state. Denne skal ha fire parametre:

Løsningsforslag:
```
def update_state(dict, state, demvotes, repvotes):
    if dict.get(state,0) != 0:
        dict[state].append((demvotes,repvotes))
    else:
        dict[state] = [(demvotes, repvotes)]
    return dict
```

Oppgave 3.4 (10%)
Skriv funksjonen read_from_file(). Den skal oppføre seg som beskrevet over.

Løsningsforslag:
```
def read_from_file():
    dikt = {}
    with open('votes.txt','r') as f:
        linje = f.readline().strip()
        while(linje):
            liste = linje.split(',')
            try:
                dikt = update_state(dikt, liste[0], int(liste[1]), int(liste[2]))
            except:
                print(f'Linje kunne ikke tolkes: {linje}')
            linje = f.readline().strip()
    return dikt
```

Oppgave 3.5 (5%)
Skriv funksjonen get_ev_for_state(state). Denne skal ta inn navnet på en stat, og *returnere* hvor mange valgmenn det er i denne staten.

Løsningsforslag:
```
def get_ev_for_state(state):
    electoral_votes = [["Arizona", 11], ["Nevada", 6], ["Pennsylvania", 20], ["Georgia", 16]]
    for el in electoral_votes:
        if el[0] == state:
            return el[1]
```

Oppgave 3.6 (5%)
Skriv funksjonen get_actual_ev(state, dempercent, reppercent).

Løsningsforslag:
```
def get_actual_ev(state, dempercent, reppercent):
    ev = get_ev_for_state(state)
    if dempercent > reppercent:
        return (ev,0)
    else:
        return (0,ev)
```

Oppgave 3.7 (5%)
Skriv funksjonen get_actual_ev_fair.

Løsningsforslag:
```
def get_actual_ev_fair(state, dempercent, reppercent):
    ev = get_ev_for_state(state)
    dem_ev = round(dempercent*ev/100)
    return (dem_ev,ev - dem_ev)
```

Oppgave 3.8 (10%)
Skriv funksjonen print_nation_results(votes).

Løsning:

```
def find_percent(state,votes):
    dem,rep = 0,0
    for el in votes[state]:
        dem += int(el[0])
        rep += int(el[1])
    prosent_dem = dem*100/(dem+rep)
    return (round(prosent_dem,2),round(100-prosent_dem,2))


def print_nation_results(votes):
    dem,rep = 0,0
    ev_norm_dem,ev_norm_rep = 0,0
    ev_fair_dem,ev_fair_rep = 0,0
    for key in votes:
        stemmer = return_total_votes(votes[key])
        dem += stemmer[0]
        rep += stemmer[1]
        prosent = find_percent(key,votes)
        pro_pr_parti = get_actual_ev(key, prosent[0], prosent[1])
        ev_norm_dem += pro_pr_parti[0]
        ev_norm_rep += pro_pr_parti[1]
        pro_pr_parti = get_actual_ev_fair(key, prosent[0], prosent[1])
        ev_fair_dem += pro_pr_parti[0]
        ev_fair_rep += pro_pr_parti[1]

    print(f'Dems got {dem} votes, reps got {rep}')
    print(f'With wta, dems got {ev_norm_dem} while reps got {ev_norm_rep} electoral votes')
    print(f'With fair, dems got {ev_fair_dem} while reps got {ev_fair_rep} electoral votes')
```

Test kode:

```
print(f"Electoral votes for Nevada should be 6: {get_ev_for_state('Nevada')}.")
print(get_actual_ev('Nevada', 50.1,49.9))
fair = get_actual_ev_fair("Nevada", 50.1, 49.9)
print(f"Hvis vi fordeler etter prosent skal begge få 3: {fair}.")
votes = read_from_file()
print_nation_results(votes)
```