Universitetet i Bergen
Det matematisk-naturvitskapelege fakultet
Institutt for informatikk

Nynorsk

Eksamen i emnet INF100
Grunnkurs i programmering
Tirsdag 28. mai 2019
Tid: 09:00 – 14:00

Tillatte hjelpemiddel: Alle trykte og skrevne
Oppgåvesettet er på 6 oppgåver og 12 sider.

Generelle råd og kommentarar:

- Les nøye gjennom oppgåvene før du byrjar å svara.
- Dersom du ikkje klarer å gi fullstendig svar på ei oppgåve, kan du likevel halda fram med dei andre. Gå ut frå at kode du skulle utvikla i oppgåver du ikkje svarer på er tilgjengeleg.
- Koden din bør vera leseleg og enkel å forstå.
- Gå ut frå at alle naudsynte import-setningar er inkluderte.
- Syns du at oppgåveteksten er uklar eller ufullstendig, må du laga dine eigne presiseringar og gi desse i svaret.
- Prosentsatsane ved kvar deloppgåve gir omtrentleg vektlegging ved sensur.
- Dei tre siste sidene i oppgåvesettet gir eit kort og ufullstendig utdrag av standardbiblioteket i Python.

Lukke til!

Dag Haugland AtleGeitung

Oppgåve 1 (5%)

Oppgåve 1a (1%) Kva tal skriv denne koden ut?

```
a = 1
print(a)
a = 3
```

Oppgåve 1b (1%) Kva tal skriv denne koden ut?

```
x = 0
def f(x=1):
    print(x)

f(2)
```

Oppgåve 1c (1%) Kva tal skriv denne koden ut?

```
def f(x):
    return x+1

a = 1
print(a + f(a))
```

Oppgåve 1d (1%) Kva tal skriv denne koden ut?

```
def f(x):
    y = x // 3
    a = x % 3
    b = a % 2
    return y + b

print(f(9) + f(10))
```

Oppgåve 1e (1%) Kva tal skriv denne koden ut?

```
def f(x, y=1):
    return x - y

x = 9
y = 4
print(f(y))
```

Oppgåve 2 (8%)

Oppgåve 2a (2%) Kva skriv denne koden ut?

```
def vurdering(x):
    if x <= 1:
        print('nei')
    if x <= 3:
        print('tvilsamt')
    if x <= 5:
        print('kanskje')
    if x <= 7:
        print('truleg')
    else:
        print('ja')

vurdering(3)
```

Oppgåve 2b (2%) Kva skriv denne koden ut?

```
def vurdering(x):
    if x <= 1:
        print('nei')
    elif x <= 3:
        print('tvilsamt')
    elif x <= 5:
        print('kanskje')
    elif x <= 7:
        print('truleg')
    else:
        print('ja')

vurdering(3)
```

Oppgåve 2c (2%) Kva skriv denne koden ut?

```
liste1 = [5, 2, 6, 9]
liste2 = []
for i in range(len(liste1)):
    verdi = liste1[i]
    if i > 0:
        verdi = i + liste1[i]/i
    liste2.append(verdi)
print(liste2)
```

Oppgåve 2d (2%) Kva skriv denne koden ut?

```
pos = ord('a')
streng = ''
while len(streng) < 3:
    streng += chr(pos)
    pos += 2
print(streng)
```

Oppgåve 3 (8%)

Oppgåve 3a (2%) Kva skriv denne koden ut?

```
def snittMedSmå(mengd):
    return mengd.intersection(range(10))

print(snittMedSmå({12, 0, 9, 10}))
```

Oppgåve 3b (2%) Kva skriv denne koden ut?

```
def f(x):
    x = x[:-1]
    x += 'sann!'
    print(x)

hallo = 'hei!'
f(hallo)
print(hallo)
```

Oppgåve 3c (2%) Kva skriv denne koden ut?

```
def f(hallo):
    for x in hallo:
        print(x[:-1] + 'sann!')
    hallo.clear()  # Fjern alle element
    hallo.add('heisann!')

hallo = {'hei!'}
f(hallo)
for x in hallo:
    print(x)
```

Oppgåve 3d (2%) Kva skriv denne koden ut?

```
x = [1, 2, 3]
y = [x, x, x]
x[1] += 1
y[2] = [4, 4, 4]
x[2] = 0
produkt = 1
for i in range(len(y)):
    produkt *= y[i][i]
print(produkt)
```

Oppgåve 4 (15%)

Oppgåve 4a (5%) Vi seier at to tal er nær kvarandre dersom ingen av dei er så mykje som dobbelt så stort som det andre. Skriv ein funksjon **`er_nær`**`(a, b)` som returnerer `True` dersom `a` og `b` er nær kvarandre, og `False` elles. **Eksempel**:

```
>>> er_nær(3, 4)
True

>>> er_nær(5, 2)
False
```

Oppgåve 4b (5%) Skriv ein funksjon **`nærliste`**`(liste)` som tar ei liste med tal som parameter, og returnerer ei ny liste. Den nye lista skal innehalda tal frå den første lista, men berre dei som er nær både første og siste tal i den første lista. **Eksempel**:

```
>>> nærliste([3, 7, 5])
[3, 5]

>>> nærliste([4, 5, 6, 7, 8])
[5, 6, 7]

>>> nærliste([7])
[7]
```

Oppgåve 4c (5%) Vi seier at eit heiltal `d` er ein faktor i eit heiltal `a` dersom `a` er deleleg med `d`. For eksempel er `3` ein faktor i `12`, men ikkje i `13`. Skriv ein funksjon **`faktor`**`(a, b)` som skriv ut alle positive heiltal som er faktor i både `a` og `b`. **Eksempel på utskrift**:

```
>>> faktor(12, 18)
Felles faktorar for 12 og 18:
1
2
3
6

>>> faktor(49, 21)
Felles faktorar for 49 og 21:
1
7

>>> faktor(4, 12)
Felles faktorar for 4 og 12:
1
2
4

>>> faktor(8, 13)
Felles faktorar for 8 og 13:
1
```

Oppgåve 5 (14%)

Oppgåve 5a (7%) I ei fotballturnering har kvart lag møtt alle dei andre laga ein gong. Vi har ei liste `lag` med namn på laga, og ei todimensjonal liste `mål` med antal mål som er blitt scora. Antal mål `lag[i]` scora mot `lag[j]` er lik `mål[i][j]`. Skriv ein funksjon **`resultat`**`(lag, mål)` som får inn dei to listene som parametrar, og som skriv ut resultata. **Eksempel på utskrift**:

```
>>> resultat(['Brann', 'Molde', 'Rosenborg', 'Viking'],
             [ [0, 3, 2, 2],
               [1, 0, 2, 4],
               [1, 0, 0, 1],
               [0, 3, 2, 0] ])
Brann - Molde 3 - 1
Brann - Rosenborg 2 - 1
Brann - Viking 2 - 0
Molde - Rosenborg 2 - 0
Molde - Viking 4 - 3
Rosenborg - Viking 1 - 2
```

Oppgåve 5b (7%) Skriv ein funksjon **`målskilnad`**`(lag, mål)` som får inn dei to listene nemnde i oppgåve a) som parametrar, og som returnerer ein oppslagstabell (ein "`dict`") med lagnamn som nøkkel og eit heiltal som verdi. Heiltalet skal vera lik differansen mellom totalt antal mål laget har scora og totalt antal mål det har sluppe inn. **Eksempel**:

```
>>> målskilnad(['Brann', 'Molde', 'Rosenborg', 'Viking'],
               [ [0, 3, 2, 2],
                 [1, 0, 2, 4],
                 [1, 0, 0, 1],
                 [0, 3, 2, 0] ])
{'Brann': 5, 'Molde': 1, 'Rosenborg': -4, 'Viking': -2}
```

Oppgåve 6 (50%)

Du skal skriva Python-kode for testing av ulike typar måleinstrument i eit laboratorium. Med jamne mellomrom kjem det inn rapport på test av eit instrument. Kvar rapport består av instrumenttype (streng), namn på laboranten som gjorde testen (streng), og om testen var vellukka (`True`/`False`). Kvar nye rapport blir registrert i ei liste `rapportar` i form av eit tuppel (instrumenttype, laborant, `True`/`False`). I tillegg blir ein oppslagstabell (ein "`dict`") `status` oppdatert. I `status` er kvar nøkkel ein instrumenttype, og kvar verdi ei liste med to heiltal. I lista `status['nano1']` er første element talet på mislukka testar utførte på instrument av typen `'nano1'`, og andre element er talet på vellukka testar på denne typen instrument. Både `rapportar` og `status` er globale variablar.

Eksempel: Vi har mottatt tre testrapportar. To av desse var på instrumenttypen `'nano1'`, der berre den første var vellukka. Mellom desse rapportane kom det rapport om ein vellukka test på instrumenttypen `'bio-x'`. Då kan datastrukturane sjå slik ut:

```
rapportar = [('nano1', 'Martha', True),
             ('bio-x', 'Martha', True),
             ('nano1', 'Dag', False)]
```

```
status = {'nano1': [1, 1], 'bio-x': [0, 1]}
```

Oppgåve 6a (7%): Skriv ein funksjon **`registrer`**`(typen, laborant, vellukka)` som tar imot data for ein ny testrapport, og oppdaterer `rapport` og `status`. Parameteren `typen` er instrumenttypen, `laboranten` er namn på laboranten, og `vellukka` seier om testen var vellukka.

Oppgåve 6b (7%): Ein instrumenttype blir godkjent dersom det er utført minst 10 testar av han, og minst 90% av testane er vellukka. Skriv ein funksjon **`godkjent`**`(typen)` som tar ein instrumenttype som parameter, og som returnerer `True` dersom instrumenttypen oppfyller desse krava, og `False` elles. **Eksempel**:

```
>>> godkjent('nano1')
False
```

Oppgåve 6c (7%): Skriv ein funksjon **`oversikt`**`()` som returnerer ein oppslagstabell (ein "`dict`") der nøklane er instrumenttypane registrerte i `status`, og verdiane er `True` for instrumenttypane som blir godkjente, og `False` for dei andre. **Eksempel**:

```
>>> oversikt()
{'nano1': False, 'bio-x': False}
```

Oppgåve 6d (7%): Skriv ein funksjon **`konkluder`**`(godkjente, underkjente)` som tar to strengar som parametrar. Funksjonen skal lagra alle godkjente instrumenttypar på ei tekstfil med namnet `godkjente`, og alle underkjente instrumenttypar på ei tekstfil med namnet `underkjente`. Første linje på filene skal vera ei av forklaringane `'Godkjente instrument'` eller `'Underkjente instrument'`. Kvar av dei andre linjene skal berre innehalda ein instrumenttype. **Eksempel på innhald i den sistnemnde fila**:

```
Underkjente instrument
nano1
bio-x
```

Oppgåve 6e (7%): Skriv ein funksjon **`plukk`**`(laborant)` som tar ein streng som parameter. Funksjonen skal returnera ei liste med alle tuppel i `rapportar` der laboranten er lik `laborant`. **Eksempel**:

```
>>> plukk('Martha')
[('nano1', 'Martha', True),
 ('bio-x', 'Martha', True)]
```

Oppgåve 6f (7%): Det viser seg at vi ikkje kan stola heilt på alle laborantene. Skriv ein funksjon `slett(laborant)` som tar ein streng som parameter. Funksjonen skal sletta alle tuppel i `rapportar` der laboranten er lik `laborant`. Vidare skal `status` oppdaterast, slik at testar utførte av `laborant` ikkje lenger blir talde med i listene i `status`. Talet på tuppel som er sletta frå `rapportar` skal returnerast. **Eksempel**:

```
>>> slett('Dag')
1
```

Nytt innhald i `rapportar` og `status`:

```
rapportar = [('nano1', 'Martha', True),
             ('bio-x', 'Martha', True)]
status = {'nano1': [0, 1], 'bio-x': [0, 1]}
```

Oppgåve 6g (8%): Vi vil kunna verifisera at konklusjonar om godkjenning og underkjenning stemmer med innhaldet i `status`. Skriv ein funksjon `verifiser(godkjente, underkjente)` som tar to strengar som parametrar. Kvar av desse strengane er namn på tekstfiler. Gå ut frå at filene eksisterer, og at dei har same format som filene nemnde i oppgåve **6d**. Funksjonen skal lesa instrumenttypane lagra på kvar av filene, og sjekka at

- alle instrumenttypar lest frå filene førekjem som nøkkel i `status`, og
- alle instrumenttypar lest frå fila med namn `godkjente` kan godkjennast, og
- ingen instrumenttypar lest frå fila med namn `underkjente` kan godkjennast.

Funksjonen skal skriva ei kort feilmelding for kvart brot på desse reglane. Funksjonen skal returnera `True` dersom alle reglane er respekterte, og `False` elles.

Ingen fleire oppgåver!

Kort og ufullstendig oversikt over nyttige operasjonar i Python:

1. Innebygde funksjonar

`len(streng)` Returnerer lengda på ein streng.
Tilsvarande for lister (`list`), mengder (`set`) og oppslagstabellar (`dict`)
`print(streng)` Skriv ein streng til skjermen.
`streng = input(tekst)` Skriv `tekst` til skjermen, og les ein streng frå tastaturet.
`pos = ord(karakter)` Returnerer talkoden til `karakter`. **Eksempel**: `ord('a')` gir `97`
`karakter = chr(pos)` Returnerer karakteren med talkode `pos`. **Eksempel**: `chr(97)` gir `'a'`
`absoluttverdi = abs(tal)` Returnerer absoluttverdien av eit tal
`minste = min(tal1, tal2)` Returnerer minste parameterverdi
`største = max(tal1, tal2)` Returnerer største parameterverdi
`range(4)` Gir tala `0, 1, 2, 3`
`range(2, 5)` Gir tala `2, 3, 4`
`range(2, 10, 3)` Gir tala `2, 5, 8`
`lesefil = open(namn, 'r')` Opnar fil med oppgitt namn for lesing, returnerer filreferanse
`skrivefil = open(namn, 'w')` Opnar fil med oppgitt namn for skriving, returnerer filreferanse

2. Reglar for divisjon

Flyttalsdivisjon: `18 / 4` gir `4.5`
Heiltalsdivisjon: `18 // 4` gir `4`
Rest ved divisjon: `18 % 4` gir `2`
Ulovlege operasjonar: `18 / 0` gir feilmelding og programkræsj.
Tilsvarande for `18 // 0` og `18 % 0`

3. Strengar

Samanskøyting: `'abc' + '123'` gir `'abc123'`
Duplisering: `'ab' * 3` gir `'abab'`
Indeksering (la $0 \leq k < $ `len(streng)`): `streng[k]` gir karakteren i posisjon `k` i `streng`
Indeksering (la $0 < k \leq$ `len(streng)`): `streng[-k]` gir karakteren i posisjon `len(streng)-k` i `streng`
Delstreng: `streng[f:t]` gir delstrengen med karakterane `streng[f],...,streng[t-1]`
Søk: `'abcabc'.index('bc')` gir `1`
(posisjonen til første førekomst av delstreng)

Sjekk på start: `'abcabc'.startswith('bc')` gir `False`
Sjekk på slutt: `'abcabc'.endsswith('cabc')` gir `True`
Sjekk på små bokstavar: `'abcabc'.islower()` gir `True`
Sjekk på store bokstavar: `'abcabc'.isupper()` gir `False`
Sjekk på siffer: `'abcabc'.isdigit()` gir `False`, `'123'.isdigit()` gir `True`

Kopi med små bokstavar: `'ab8XY'.lower()` gir `'ab8xy'`
Kopi med store bokstavar: `'ab8XY'.upper()` gir `'AB8XY'`

Fjern blanke i starten og på slutten: `'  abc abc '.strip()` gir `'abc abc'`
Ulovleg operasjon: `streng[0]='x'` Forsøk på å gi ny verdi i oppgitt posisjon gir feilmelding og programkræsj.
Oppdeling: `'abc 123 xyz'.split()` gir lista `['abc', '123', 'xyz']`

4. **Lister**

Duplisering, indeksering, dellister, og søk: Som for strengar.
Oppretting av tom liste:
```
liste = []
```
Oppretting:
```
liste = [7, 4, 8]
```
Gi ny verdi i gitt posisjon:
```
liste[posisjon] = verdi
```
Fjernar og returnerer element i gitt posisjon:
```
verdi = liste.pop(posisjon)
```
Fjernar og returnerer element i siste posisjon:
```
verdi = liste.pop()
```
Legg til element på slutten:
```
liste.append(verdi)
```
Legg til element før oppgitt posisjon:
```
liste.insert(posisjon, verdi)
```

5. **Mengder**

Oppretting av tom mengd:
```
mengd = set()
```
Oppretting:
```
mengd = {7, 4, 8}
```
Legg til element:
```
mengd.add(element)
```
Fjern oppgitt element:
```
mengd.discard(element)
```
Fjern alle element:
```
mengd.clear()
```
Returner differansen med ei anna mengd:
```
mengd1.difference(mengd2)
```
Returner unionen med ei anna mengd:
```
mengd1.union(mengd2)
```
Returner snittet med ei anna mengd:
```
mengd1.intersection(mengd2)
```

6. **Tuppel**

Oppretting: `tuppel = ('abc', 4, True)`
Indeksering: `tal = tuppel[1]`
Ulovleg operasjon: `tuppel[2] = False` Forsøk på å gi ny verdi til oppgitt element gir feilmelding og programkræsj.

6. **Oppslagstabellar (også kalla "`dict`")**

Oppretting: `tabell = {'gulrot': 'grønnsak', 'mango': 'frukt'}`
Oppretting av tom tabell: `tabell = {}`
Slå opp i tabellen: `verdi = tabell[nøkkel]`
Slå opp i tabellen, returnerer standardverdi dersom nøkkelen ikkje finst:
`verdi = tabell.get(nøkkel, standard)`
Eksempel: `verdi = tabell.get('banan', 'ukjent')`
Legg til nytt eller oppdater eksisterande oppslag:
`tabell[nøkkel] = verdi`
Fjern oppslag: `verdi = tabell.pop(nøkkel)`
Fjern alle oppslag: `tabell.clear()`
Returner liste med nøklar i tabellen: `tabell.keys()`
Returner liste med verdiar i tabellen: `tabell.values()`

7. **Filer**
Les fram til neste linjeskift, returner linja som ein streng: `streng = lesefil.readline()`
Les fram til filslutt, returner innhaldet som ein streng: `streng = lesefil.read()`
Les fram til filslutt, returner innhaldet som ei liste med ein streng for kvar leste linje:
`liste = lesefil.readlines()`
Skriv ein streng til fila: `skrivefil.write(streng)`
Lukk fila: `fil.close()`