Prøveeksamen 2017

1(a)

Oppgave 1a)

Hva er verdien til **tall** etter at følgende kode er utført?

```
tall = 4+(3*2)
tall = tall-1
```

9.0

1(b)

Oppgave 1b)

Hva er verdien til **j** etter at følgende kode er utført?

```
i = 1
j = 2
while i < 4:
  j = j * i
  i = i + 1
```

12.0

1(c)

Oppgave 1c)

Hva skrives ut på skjermen når følgende kode utføres?

```
tall = [2, 6, 3, 6, 9]
a = 0
b = 0
i = 0

while i < len(tall):
  if tall[i] > a:
    b = b + tall[i]
    a = tall[i]
  i = i + 1

print(b)
```

17.0 .

1(d)

Oppgave 1d)

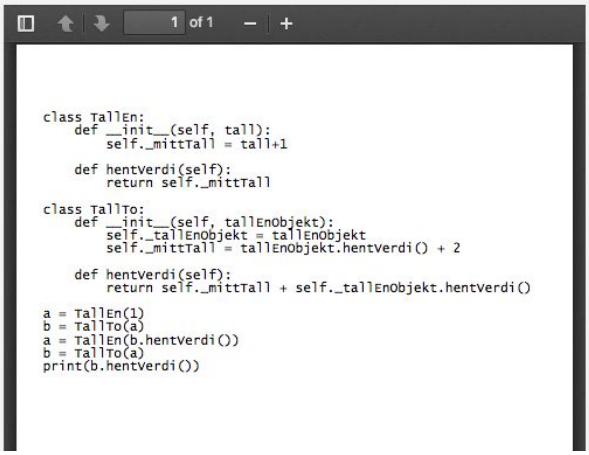
Vi har en funksjon **repeter** som vist nedenfor:

```
def repeter(a):
  a = a + a
  return a
```

Hva skrives ut på skjermen når følgende kode utføres?

```
a = "ab"
b = repeter(a)
print(a+b)
```

ababab



```
class TallEn:
    def __init__(self, tall):
        self._mittTall = tall+1

    def hentVerdi(self):
        return self._mittTall

class TallTo:
    def __init__(self, tallEnObjekt):
        self._tallEnObjekt = tallEnObjekt
        self._mittTall = tallEnObjekt.hentVerdi() + 2

    def hentVerdi(self):
        return self._mittTall + self._tallEnObjekt.hentVerdi()

a = TallEn(1)
b = TallTo(a)
a = TallEn(b.hentVerdi())
b = TallTo(a)
print(b.hentVerdi())
```

1(e) **Oppgave 1e)**

Hva skrives ut på skjermen når koden i pdf-vedlegget utføres?

16

1(f) **Oppgave 1f)**

Gitt en funksjon **voks** som vist her:

```
def voks(alder):
  alder = alder +1
```

Hva skrives ut på skjermen når følgende kode utføres?

```
pers_alder = 29
voks(pers_alder)
print(pers_alder)
```

29.0

1(g)

Oppgave 1g)

Gitt en funksjon **brillesjekk** som vist her:

```
def brillesjekk(styrke):
  ny_styrke = [2.5, 2.75]
  styrke = ny_styrke
```

Hva skrives ut på skjermen når koden nedenfor utføres?

```
pers_styrke = [1.5, 1.5]
brillesjekk(pers_styrke)
print(pers_styrke[0])
```

1.5 .

1(h)

Oppgave 1h)

Gitt en funksjon **brillesjekk2** som vist her:

```
def brillesjekk2(styrke):
  styrke[0] = 1.75
```

Hva skrives ut på skjermen når følgende kode utføres?

```
pers_styrke = [1.5, 1.5]
brillesjekk2(pers_styrke)
print(pers_styrke[0])
```

1.75 .

1(i)

Oppgave 1i)

Hva skrives ut når følgende programsetninger kjøres?

```
a = [1, 2, 3]
b = a
b[0] += 1
print(a)
```

Velg ett alternativ

- (•) [2,2,3]
- ( ) [1,2,3]
- ( ) [1,1,2,3]

1(j)

Oppgave 1j)

liste = [[5,4], [9,12,3]]

a) Hva er verdien av liste[1][0]?

9

b) Hva er verdien av liste[0]?

[5,4]

1(k)

Oppgave 1k)

ordbok = { "b":[4,3,5], "a":[0] }

Hva er verdien av ordbok["a"][0]?

0

2(a)

Oppgave 2a)

Hva er galt i følgende kode? (kort forklaring holder - én setning er gjerne nok)

```
def gang_med_to(tall):
    return tall*2

svar = gang_med_to(5,4)
```

Skriv ditt svar her...

Format

Funksjonskallet sender med to parametre, mens funksjonen skal kun ta imot ett argument.

2(b)

Oppgave 2b)

Hva er galt i følgende kode? (kort forklaring holder - én setning er gjerne nok)

```
def hent_pris(alder):
    if alder<18:
        return print(100)
    else:
        return print(200)

antall = 3
pris = hent_pris(18)
totalt = antall*pris
```

Skriv ditt svar her...

Format

Funksjonen returnerer print(200). Print gir ingen returverdi, så det som lagres i "pris" er en "Nonetype". Videre når vi skal gange antall (som er en int) med pris (som er "None") så får vi en feilmelding på at multiplikasjon med disse to typene ikke er mulig.

Oppgave3(a)Oppgave 3a)

Skriv en funksjon **hastighet(fart)** som skal returnere en tekst-streng basert på heltallsverdien (verdi av type int) i parameteren fart. Parameteren **fart** er ment å angi den målte farten til en bil i en 60-sone. Dersom **fart** er 60 eller mindre, skal funksjonen returnere en streng (verdi av type str) som består av "fart:" og den målte farten. F.eks. skal kallet **hastighet(56)** returnere strengen "fart:56". Dersom **fart** er høyere enn 60, skal funksjonen returnere strengen "fart:over 60". Kallene **hastighet(61)** og **hastighet(100)** skal altså begge returnere strengen "fart:over 60".

```
def hastighet(fart):
```

```
    if fart <= 60:
        return "fart:" + str(fart)
    else:
        return "fart: over 60"
```

Oppgave 3b)

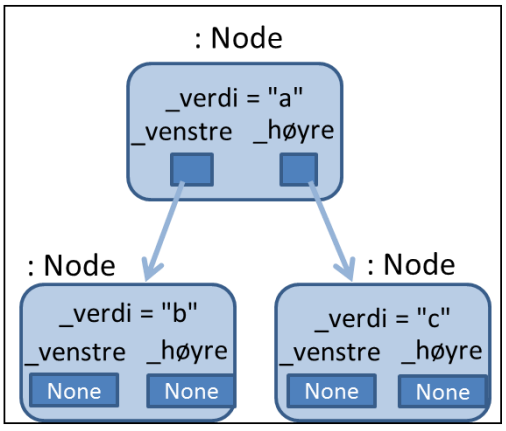
a) Skriv en funksjon **sjekkVerdier(tallene, min, max)** hvor **tallene** er en liste av heltallsverdier (liste av verdier av type int), mens **min** og **max** er heltall (verdi av type int). Funksjonen skal sjekke om alle verdiene i lista **tallene** er ekte større (ikke lik) enn **min** og ekte mindre (ikke lik) enn **max**. Dersom alle verdiene er innenfor dette intervallet skal metoden returnere **True**, ellers skal metoden returnere **False**.

b) Beskriv i en kommentar nederst i programkoden hvordan metoden din oppfører seg dersom **min > max**.

```
def sjekkVerdier(tallene, min, max):
    for tall in tallene:
        if tall <= min or tall => max:
            return False
    return True


Dersom min > max vil testen "tall <= min or tall => max" alltid være sant,
så funksjonen vil alltid returnere False.
```

Oppgave 3c)

```
class Node :
   def __init__(self, innhold) :
      self._verdi = innhold
      self._venstre = None
      self._hoyre = None

   def settInnHoyre (self, h) :
      self._hoyre = h

   def settInnVenstre (self, v) :
      self._venstre = v
```



Gitt klassen Node som vist i vedlagte kode, skriv en funksjon **hovedprogram()** som oppretter 3 objekter med verdiene "a", "b" og "c" i en struktur som vist i figuren. Du kan anta at klassen Node er importert til programmet ditt.
Skriv ditt svar her...

```
def hovedprogram():
    a = Node("a")
    a.settInnVenstre(Node("b"))
    a.settInnHoyre(Node("c"))
```

Oppgave 4 a)

I oppgave 4 a-g (Klasser og objekter) skal du skrive deler av et program for et elektronisk bruktmarked. Den samlede teksten for hele oppgave 4 (a-g) er lagt ved hver deloppgave.

Oppgave 4 a

Skriv klassen Bud med alle metoder som spesifisert i vedlagt dokument.
(Senere deloppgaver ber om andre deler av koden for det elektroniske bruktmarkedet.)

```
#filnavn: bud.py

class Bud:
    def __init__(self, budgiver, budStr):
        self._budgiver = budgiver
        self._budStr = budStr
        if budStr < 0:
            self._budStr = 1

    def hentBudgiver(self):
        return self._budgiver

    def hentBudStr(self):
        return self._budStr
```

Oppgave 4b)

(fortsettelse oppgave 4) Klasser og objekter)

Oppgave 4 b
Skriv klassen Annonse med alle metoder.

```python
#filnavn: annonse.py
from bud import Bud

class Annonse:
    def __init__(self, annTekst):
        self._annTekst = annTekst
        self._budListe = []

    def hentTekst(self):
        return self._annTekst

    def giBud(self, hvem, hva):
        nyttBud = Bud(hvem, hva)
        self._budListe.append(nyttBud)


    def antBud(self):
        return len(self._budListe)

    def hoyesteBud(self):
        hoyeste = None
        hoyesteVerdi = 0
        for bud in self._budListe:
            if bud.hentBudStr() > hoyesteVerdi:
                hoyeste = bud
                hoyesteVerdi = bud.hentBudStr()

        return hoyeste
```

Oppgave 4c)

(fortsettelse oppgave Klasser og objekter)

Oppgave 4 c
Skriv klassen Kategori med alle metoder, som spesifisert i vedlagt dokument.

```python
#filnavn: kategori.py
from annonse import Annonse

class Kategori:
```

```
    def __init__(self, katNavn):
        self._katNavn = katNavn
        self._annonseListe = []

    def nyAnnonse(self, annTekst):
        nyAnn = Annonse(annTekst)
        self._annonseListe.append(nyAnn)

    def hentAnnonser(self):
        return self._annonseListe
```

Oppgave 4d)

(fortsettelse oppgave Klasser og objekter)

Oppgave 4 d

Skriv klassen Bruktmarked med alle metoder og representasjon som spesifisert i vedlagt dokument.

```
#filnavn: bruktmarked.py
from kategori import Kategori

class BruktMarked:
    def __init__(self):
        self._kategorier = {}

    def nyKategori(self, katNavn):
        #sjekker om kategorien finnes fra foer, hvis den ikke gjoer det
oppretter vi en ny kategori
        if finnKategori(katNavn) == None:
            nyKat = Kategori(katNavn)
            self._kategorier[katNavn] = nyKat
            return nyKat
        else:
            return None

        """ alternativer:
        if not finnKategori(katNavn): #siden None er False.
        ogsaa droppe else siden den vil returnere dersom ny kategori skal
opprettes.
        """
```

```
    def finnKategori(self, katNavn):
        for kat in self._kategorier:
            if kat == katNavn:
                return self.kategrier[kat]
        return None
```

Oppgave 4e)

(fortsettelse oppgave Klasser og objekter)

Oppgave 4 e
Skriv metoden **kraftBud** i klassen **Annonse**, som spesifisert i vedlagt dokument.

```
    def kraftBud(self, hvem, belop, maxV):
        budBelop = belop

        #henter ut hoyeste buds storrelse
        hoyeste = self.hoyesteBud().hentBudStr()
        #dersom belop er mindre enn hoyeste, settes vaart budbelop til
hoyeste +1
        if belop < hoyeste:
            budBelop = hoyeste + 1

        #dersom hoyeste +1 er storre enn max, settes max til verdien i
stede.
        if budBelop > maxV:
            budBelop = maxV

        #oppretter belopet og setter inn bud ved aa kalle paa "giBud()"
        self.giBud(hvem, budBelop)
```

Oppgave 4f)

(fortsettelse oppgave Klasser og objekter)

Oppgave 4 f
Skriv et hovedprogram som bruker klassene fra tidligere deloppgaver slik det er spesifisert i vedlagt dokument.

```python
def hovedprogram():
    marked = BruktMarked()
    #metoden nyKategori returnerer den nye kategorien vaar
    kat = marked.nyKategori("sykkellykt")
    ann = kat.nyAnnonse("New York sykkellykt")
    ann.giBud("Peter", 42)
    ann.giBud("Ann", 0)
    ann.kraftBud("Mary", 40, 50)

    hoyesteBudStr = ann.hoyesteBud().hentBudStr()
    budGiver = ann.hoyesteBud().hentBudgiver()

    #sjekker at utskriften gir 43 gitt av Mary
    print(hoyesteBudStr, "gitt av", budGiver )

    #alternativt med assert:
    assert hoyesteBudStr == 43
    assert budGiver == "Mary"

hovedprogram()
```

Oppgave4(g)Oppgave 4g)

(fortsettelse oppgave Klasser og objekter)

Oppgave 4 g

Skriv metoden **tellLaveBud** i klassen **Bruktmarked** slik det er spesifisert i vedlagt dokument.

```python
#i klassen bruktmarked
def tellLaveBud(self):
    antall = 0
    for kat in self._kategorier:
        antall += self._kategorier[kat].tellLaveBud()

    return antall


#i klassen Annonse:
def tellLaveBud(self):
    hoyeste_hitill = 0
```

```
    antall_lavere = 0
    for bud in self._budListe:
        if bud.hentBudStr() > hoyeste_hitill:
            hoyeste_hitill = bud.hentBudStr():
        else :
            antall_lavere += 1

    return antall_lavere

#i klassen kategori:
def tellLaveBud(self):
    antall = 0
    for ann in self._annonseListe
        antall += ann.tellLaveBud()

    return antall
```

Oppgave 5

1) Skriv en funksjon **arverekke(forfader, etterkommer, forstefodte)**
som kan brukes for å returnere en liste med alle navn i arverekken fra og med **forfader** (av type str) til og med **etterkommer** (av type str) dersom denne kan utledes fra mappingen **forstefodte**. Parameteren **forstefodte** kan antas å være en ordbok (dict) fra foreldre til førstefødte barn. Når man slår opp med et navn (av type str) som nøkkel, får man altså navnet (av type str) på det førstefødte barnet som verdi (dersom denne eksisterer). Personene lagret i **forstefodte** danner ikke nødvendigvis en sammenhengende arverekke.
Dersom **forfader** og **etterkommer** ikke hører sammen i en felles arverekke, skal funksjonen returnere en tom liste.
Ved bruk av funksjonen, skal følgende kodesekvens i Python resultere i at verdien til **personer** blir en liste med navnene "Halfdan","Harald","Eirik":
barn = {"Halfdan":"Harald", "Christian":"Hans", "Harald":"Eirik"} personer = arverekke("Halfdan", "Eirik", barn)

2) Beskriv, i en kommentar i funksjonen, hvordan funksjonen din oppfører seg dersom flere av personene i arverekken har nøyaktig samme navn.

```
#beklager denne ble litt rotete..

def arverekke(forfader, etterkommer, forstefodte):
    liste = []
```

```
    liste.append(forfader)
    neste = forstefodte[forfader]
    while neste != None:
        liste.append(neste) #legger til neste
        if neste == etterkommer: #hvis vi finner etterkommeren returnerer
vi listen
            return liste

        #ellers flytter vi oss videre til neste. Sjekker forst om den
finnes
        if neste in forstefodte.keys():
            neste = forstefodte[neste] #oppdaterer nestevariablen
        else:
            #hvis den ikke finnes saa returnerer vi tom liste fordi da fant
vi ikke rekkefoelgen.
            return []

    #hvis vi aldri finner etterkommeren returnerer vi tom liste
    return []

barn = {"Halfdan":"Harald", "Christian": "Hans", "Harald":"Eirik"}
personer = arverekke("Halfdan", "Per", barn)
```