IT2 - Tentamen høsten 2022

Del 1 - uten hjelpemidler

Navn: _____

Oppgave 1

1.1

```
1 tall = (5*5)-4
2 tall = tall-1
```

Hva er verdien til tall ?

Svar: _____

1.2

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

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

Svar: _____

1.3

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

a. Hva er verdien av `liste[1][0]`?

   Svar: _____

b. Hva er verdien av `liste[0]`?

   Svar: _____

1.4

Hva skrives ut her?

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

a = "ab"
b = repeter(a)

print(a + b)
```

Svar: _____

1.5

Hva skrives ut her?

```
a = 10
b = 1
while a > 0:
  b = b * 2
  a = a - b
print("a = " + a)
print("b = " + b)
```

Svar: _____

1.6

Hva skrives ut her?

```
serie = "0"
for i in range(5,10):
    serie = serie + str(i)
print("serie = " + serie)
```

Svar: _____

1.7

Hva skrives ut her?

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

- ☐ `[1, 2, 3]`
- ☐ `[1, 1, 2, 3]`
- ☐ `[2, 2, 3]`

Oppgave 2

2.1

Hva skrives ut her?

```
class Bil:
  def __init__(self, merke, modell):
    self.merke = merke
    self.modell = modell
    self.km = 0

  def kjør(self, antall):
    self.km += antall

  def hent_km(self):
    return self.km

thors_bil = Bil("Toyota", "Corolla")
thors_bil.kjør(500)
print(thors_bil.hent_km)
```

Svar: _____

2.2

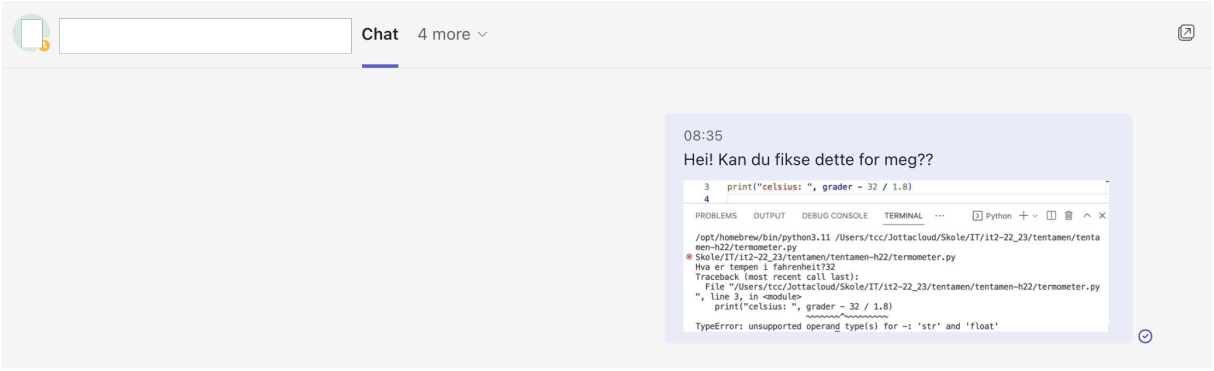
Hva skrives ut her? (klassen `Bil` er lik som i oppgave 2.1)

```python
class Bil:
  def __init__(self, merke, modell):
    self.merke = merke
    self.modell = modell
    self.km = 0

  def kjor(self, antall):
    self.km += antall

  def hent_km(self):
    return self.km

thors_bil = Bil("Toyota", "Corolla")
thors_bil.kjor(500)
ravis_bil = Bil("Hyundai", "Ioniq 5")
thors_bil = ravis_bil
thors_bil.kjor(200)
ravis_bil.kjor(100)
print(thors_bil.hent_km)
```

Svar: _____

2.3

Du har fått denne meldingden av bestevennen din.




```
3   print("celsius: ", grader - 32 / 1.8)
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ··· Python

```
/opt/homebrew/bin/python3.11 /Users/tcc/Jottacloud/Skole/IT/it2-22_23/tentamen/tentamen-h22/termometer.py
Skole/IT/it2-22_23/tentamen/tentamen-h22/termometer.py
Hva er tempen i fahrenheit?32
Traceback (most recent call last):
  File "/Users/tcc/Jottacloud/Skole/IT/it2-22_23/tentamen/tentamen-h22/termometer.py", line 3, in <module>
    print("celsius: ", grader - 32 / 1.8)
                       ~~~~~~~^~~~~~~~~~
TypeError: unsupported operand type(s) for -: 'str' and 'float'
```

Hva svarer du?

Svar: _____

Del 2 - Med hjelpemidler

Oppgave 3 - Viken-brukere

I denne oppgaven skal du lage funksjoner som IT-avdelingen i Viken kan bruke for å generere brukernavn og eposter.

3.1

Lag en funksjon `lag_brukernavn` som tar inn et fullt navn (f.eks. "Jo Brochmann") og returnerer et brukernavn. Brukernavnet skal bestå av personens fornavn, etterfulgt av den første bokstaven i etternavnet. Alle bokstavene skal være små bokstaver. For eksempel skal `lag_brukernavn("Jo Brochmann")` returnere `"job"`.

Du kan anta at alle navn kun består av en fornavn og et etternavn.

> HINT: `min_tekst.split()` kan være nyttig.
> HINT: `min_tekst = min_tekst.lower()` gjør at `min_tekst` kun består av små bokstaver.

3.2

Lag en funksjon `lag_epost` som tar inn et navn og en skole som argumenter, og returnerer en passende epost. Epost-adressen skal bestå av brukernavnet (samme regler som i 3.1), en `@`, skolenavnet og til slutt `.viken.no`. For eksempel skal `lag_epost("Thor Coward", "Sandvika")` returnere `thorc@sandvika.viken.no`.

3.3

Lag en funksjon `fjern_falske_brukere` som tar inn en liste med brukernavn og et nøkkelord, og returnerer en liste med brukernavn som ikke inneholder nøkkelordet. For eksempel skal `fjern_falske_brukere(["thorc", "ravim", "ceiliet", "fredrikg"], "fred")` returnere `["thorc", "ravim", "ceciliet"]`

Oppgave 4 - Fotball

I denne oppgaven skal du lage et program som holder oversikt over fotballspillere, lag og serier.

4.1 - Modell og oppsett

Les gjennom oppgavetesktene i oppgave 4.2, 4.3 og 4.4

1. Tegn et UML-diagram med klassene og metodene til programmet. Bruk diagrams.net eller tegn for hånd.
2. Lag et skall i Python med klassene og metodene. For eksempel slik:

```python
class Eksempel:
    def __init__(self):
        self._eksempelvariabel1
        self._eksempelvariabel2

    def eksempelmetode1(self):
        pass

    def eksempelmetode2(self):
        pass
```

4.2 - Klassen Spiller

Skriv klassen `Spiller` i filen `spiller.py` med følgende grensesnitt.

Klassen skal ha en konstruktør med parameter for `navn` (i tillegg til `self`). Konstruktøren skal opprette instansvariablene `_navn`, som får verdi fra parameteren `navn`, og `_antall_kamper` som skal settes til heltallet 0. I tillegg til konstruktøren skal klassen ha følgende metoder:

- `hent_navn` som returnerer `_navn`
- `hent_antall_kamper` som returnerer `_antall_kamper`
- `spill_kamp` som øker `_antall_kamper` med 1
- `sjekk_navn` som tar inn et navn som parameter, og returnerer `True` hvis spillerens navn inneholder ett av navnene.

  For eksempel for spillernavnet “Caroline Graham Hansen” skal `sjekk_navn("Caroline")`, `sjekk_navn("Hansen")` og `sjekk_navn("Caroline Jenssen")` alle returnere `True`, mens `sjekk_navn("Karoline")` skal returnere `False`.

4.3 - Klassen Lag

Skriv klassen `Lag` i filen `lag.py` med følgende grensesnitt.

Klassen skal ha en konstruktør med parameter for `navn` (i tillegg til `self`). Konstruktøren skal opprette instansvariablene `_navn`, som får verdi fra parameteren `navn`, `_spillere` som er en tom liste, `_seier` som skal settes til heltallet 0, `_uavgjort` som skal settes til heltallet 0 og `_tap` som skal settes til heltallet 0. I tillegg til konstruktøren skal klassen ha følgende metoder:

- `hent_navn` som returnerer `_navn`
- `hent_poeng` som returnerer antall poeng laget har, hver seier gir 3 poeng, uavgjort gir 1 poeng og tap gir 0 poeng.

  For eksempel skal `hent_poeng()` returnere 4 for et lag med 1 seier, 1 uavgjort og 2 tap.
- `legg_til_spiller` som tar inn en spiller som parameter og legger den til i listen `_spillere`
- `spill_kamp` som øker antall kamper for alle spillerne på laget med 1
- `seier` som øker `_seier` med 1 og kjører metoden `spill_kamp` (slik: `self.spill_kamp()`)
- `uavgjort` som øker `_uavgjort` med 1 og kjører metoden `spill_kamp`
- `tap` som øker `_tap` med 1 og kjører metoden `spill_kamp`
- `finn_spiller` som tar inn et navn som parameter, hvis en spiller på laget har navnet i navnet sitt, returnerer den spilleren, ellers returnerer den `None` (Samme regler gjelder for sjekk av navn som i metoden `sjekk_navn` i klassen `Spiller`).

4.4 - Klassen Serie

Skriv klassen `Serie` i filen `serie.py` med følgende grensesnitt.

Klassen skal ha en konstruktør med parameter for `navn` (i tillegg til `self`). Konstruktøren skal opprette instansvariablene `_navn`, som får verdi fra parameteren `navn` og `_lag` som er en tom liste. I tillegg til konstruktøren skal klassen ha følgende metoder:

- `hent_navn` som returnerer `_navn`
- `hent_lagliste` som returnerer `_lag`.
- `legg_til_lag` som tar inn et lag som parameter og legger den til i listen `_lag`
- `spill_kamp` som tar inn parameterne `hjemmelag` som er et lag-objekt, `bortelag` som er et lag-objekt, `hjemmemaal` som er et heltall og `bortemaal` som er et heltall, hvis et lag har flere mål enn det andre, skal laget med flest mål vinne og det andre laget tape, hvis lagene har like mange mål, skal begge lagene spille uavgjort.

- `finn_spiller` som tar inn et navn som parameter, hvis en spiller i serien har navnet i navnet sitt, returnerer den spilleren, ellers returnerer den `None` (Samme regler gjelder for sjekk av navn som i metoden `sjekk_navn` i klassen `Spiller`).