UNIVERSITETET I BERGEN

KANDIDAT

112

PRØVE

INFO132 0 Innføring i programmering

Emnekode	INFO132
Vurderingsform	Skriftlig eksamen
Starttid	19.12.2023 08:00
Sluttid	19.12.2023 12:00
Sensurfrist	--
PDF opprettet	21.12.2023 12:11

Oppgave 1: Flervalg

Oppgave	Tittel	Oppgavetype
i	INFO132: Informasjon om oppgave 1	Informasjon eller ressurser
1	INFO132: Flervalg - konvertering	Flervalg
2	INFO132: Flervalg - sammenligning	Flervalg
3	INFO132: Flervalg - syntaks	Flervalg
4	INFO132: Flervalg - litteral	Flervalg
5	INFO132: Flervalg - uttrykk	Flervalg
6	INFO132: Flervalg - uttrykk	Flervalg
7	INFO132: Flervalg - uttrykk	Flervalg
8	INFO132 Flervalg - tuppel	Flervalg
9	INFO132: Flervalg - liste	Flervalg
10	INFO132: Flervalg - filer	Flervalg
11	INFO132: Flervalg - tegnstreng	Flervalg
12	INFO132: Flervalg - in	Flervalg
13	INFO132: Flervalg - sammenligning	Flervalg
14	INFO132: Flervalg - sammenligning	Flervalg
15	INFO132: Flervalg - sammenligning	Flervalg
16	INFO132: Flervalg - sammenligning	Flervalg
17	INFO132: Flervalg - sammenligning	Flervalg
18	INFO132: Flervalg - type	Flervalg
19	INFO132: Flervalg - type	Flervalg

20	INFO132: Flervalg - intervall	Flervalg
21	INFO132: Flervalg - fortegnelse	Flervalg
22	INFO132: Flervalg - mengder	Flervalg
23	INFO132: Flervalg - fortegnelse	Flervalg
24	INFO132: Flervalg - fortegnelse	Flervalg
25	INFO132: Flervalg - listebygger	Flervalg
26	INFO132: Flervalg - listebygger	Flervalg
27	INFO132: Flervalg - formattering	Flervalg
28	INFO132: Flervalg - parameter	Flervalg
29	INFO132: Flervalg - GUI	Flervalg
30	INFO132: Flervalg - setninger og uttrykk	Flervalg
31	INFO132: Flervalg - funksjoner	Flervalg
32	INFO132: Flervalg - funksjoner	Flervalg
33	INFO132: Flervalg - reserverte ord	Flervalg
34	INFO132: Flervalg - inndata	Flervalg
35	INFO132: Flervalg - navnerom	Flervalg
36	INFO132: Flervalg - innebygd funksjon	Flervalg
37	INFO132: Flervalg - setninger og uttrykk	Flervalg
38	INFO132: Flervalg - setninger og uttrykk	Flervalg
39	INFO132: Flervalg - presedens	Flervalg
40	INFO132: Flervalg - presedens	Flervalg
41	INFO132: Flervalg - presedens	Flervalg

42	INFO132: Flervalg - overbelastning	Flervalg
43	INFO132: Flervalg - print	Flervalg
44	INFO132: Flervalg - break	Flervalg
45	INFO132: Flervalg - continue	Flervalg
46	INFO132: Flervalg - global	Flervalg
47	INFO132: Flervalg - mengder	Flervalg
48	INFO132: Flervalg - GUI	Flervalg
49	INFO132: Flervalg - GUI	Flervalg
50	INFO132: Flervalg - variabler	Flervalg
51	INFO132: Flervalg - klasser	Flervalg
52	INFO132: Flervalg - GUI	Flervalg
53	INFO132: Flervalg - filer	Flervalg
54	INFO132: Flervalg - filer	Flervalg
55	INFO132: Flervalg - filer	Flervalg
56	INFO132: Flervalg - while	Flervalg
57	INFO132: Flervalg - parametre	Flervalg
58	INFO132: Flervalg - bygging	Flervalg
59	INFO132: Flervalg - bygging	Flervalg
60	INFO132: Flervalg - fortegnelse	Flervalg

Oppgave 2: Programforståelse

Oppgave	Tittel	Oppgavetype
i	INFO132: Informasjon om oppgave 2	Informasjon eller ressurser

61	INFO132: Programforståelse	Flervalg
62	INFO132: Programforståelse	Flervalg
63	INFO132: Programforståelse	Flervalg
64	INFO132: Programforståelse	Flervalg
65	INFO132: Programforståelse	Flervalg
66	INFO132: Programforståelse	Flervalg
67	INFO132: Programforståelse	Flervalg
68	INFO132: Programforståelse	Flervalg
69	INFO132: Programforståelse	Flervalg
70	INFO132: Programforståelse	Flervalg

Oppgave 3: Programutfylling

Oppgave	Tittel	Oppgavetype
ℹ	INFO132: Informasjon om oppgave 3	Informasjon eller ressurser
71	INFO132: Programutfylling	Flervalg
72	INFO132: Programutfylling	Flervalg
73	INFO132: Programutfylling	Flervalg
74	INFO132: Programutfylling	Flervalg
75	INFO132: Programutfylling	Flervalg
76	INFO132: Programutfylling	Flervalg
77	INFO132: Programutfylling	Flervalg
78	INFO132: Programutfylling	Flervalg
79	INFO132: Programutfylling	Flervalg

80	INFO132: Programutfylling	Flervalg

Oppgave 4: Programmering

Oppgave	Tittel	Oppgavetype
i	INFO132: Informasjon om oppgave 4	Informasjon eller ressurser
81	INFO132: Programmering	Programmering
82	INFO132: Programmering	Programmering
83	INFO132: Programmering	Programmering
84	INFO132: Programmering	Programmering
85	INFO132: Programmering	Programmering
86	INFO132: Programmering	Programmering
87	INFO132: Programmering	Programmering
88	INFO132: Programmering	Programmering
89	INFO132: Programmering	Programmering
90	INFO132: Programmering	Programmering

Kommentarer

Oppgave	Tittel	Oppgavetype
91	INFO132: Kommentarer til eksamen	Langsvar

1 INFO132: Flervalg - konvertering

Hva er resultatet av **int('2.71828')**?
Velg ett alternativ:

- ValueError: ...
- 3
- 2
- 2.71828

2 INFO132: Flervalg - sammenligning

i = 64; j = 82** . Nå er **i** og **j**
Velg ett alternativ:

- Identiske, men ikke ekvivalente
- Ekvivalente og identiske
- Ekvivalente, men aldri identiske
- Hverken ekvivalente eller identiske

3 INFO132: Flervalg - syntaks

I Python kan **2023** *ikke* være
Velg ett alternativ:

- Et objekt
- Et litteral
- Et variabelnavn
- En verdi

4 INFO132: Flervalg - litteral

I Python er dette:
3.14159
Velg ett alternativ:

- en tegnstreng
- en variabel
- et heltall
- et flyttal / desimaltall

5 INFO132: Flervalg - uttrykk

int(3*'3' + 2*2*'2')
Velg ett alternativ:

- 3332222
- 944
- 17
- SyntaxError: ...

6 INFO132: Flervalg - uttrykk

17 % 4 gir som resultat
Velg ett alternativ:

- 1
- 4.0
- 4.25
- 4

7 INFO132: Flervalg - uttrykk

13 // 4 gir som resultat
Velg ett alternativ:

- 3.0
- 1
- 3.25
- 3

8 INFO132 Flervalg - tuppel

(1, 2, 4).remove(3) gir som resultat
Velg ett alternativ:

- None
- AttributeError
- (1, 2, 4)
- ValueError: ...

9 INFO132: Flervalg - liste

[1, 3, 4].append(2) gir som resultat
Velg ett alternativ:

- AttributeError: ...
- 2
- None
- [1, 3, 4, 2]

10 INFO132: Flervalg - filer

Anta at **fil** er åpnet for lesning. Hva skrives ut her?

```
i = 0
for x in fil.readlines():
    i += 1
print( i )
```

Velg ett alternativ:

- Skriver ut antall linjer i fil
- Skriver ut antall tegn i fil
- Syntax Error: ... fil.readlines() has no argument
- 1

11 INFO132: Flervalg - tegnstreng

'Programmering'[: 1 : -3] gir som resultat
Velg ett alternativ:

- 'grmg'
- 'rogrammer'
- 'grmgP'
- ''

12 INFO132: Flervalg - in

'arb' in 'Abrakadabra' gir som resultat
Velg ett alternativ:

- True
- ValueError: ...
- False
- None

13 INFO132: Flervalg - sammenligning

list('Abkrakadabra').sort() == ['A', 'b', 'k', 'r', 'a', 'k', 'a', 'd', 'a', 'b', 'r', 'a'] gir resultat
Velg ett alternativ:

- SyntaxError: ...
- ValueError: ...
- True
- False

14 INFO132: Flervalg - sammenligning

sorted(set('Abkrakadabra')) gir resultat
Velg ett alternativ:

- {'Abrakadabra'}
- ['A', 'a', 'b', 'd', 'k', 'r']
- ['A', 'a', 'a', 'a', 'a', 'b', 'b', 'd', 'k', 'k', 'r', 'r']
- {'A', 'a', 'b', 'd', 'k', 'r'}

15 INFO132: Flervalg - sammenligning

listeA = ['Bergen', 'Oslo', 'Trondheim', 'Tromsø']
listeB = listeA.copy()
Nå er **l1** og **l2**
Velg ett alternativ:

- Ekvivalente, og kanskje identiske
- Både ekvivalente og identiske
- Ekvivalente, og ikke identiske
- Hverken ekvivalente eller identiske

16 INFO132: Flervalg - sammenligning

x == y vil si at **x** og **y** er
Velg ett alternativ:

- ekvivalente og ikke identiske
- ekvivalente men ikke nødvendigvis identiske
- ekvivalente og identiske
- identiske men ikke nødvendigvis ekvivalente

17 INFO132: Flervalg - sammenligning

x is y vil si at **x** og **y** er
Velg ett alternativ:

- identiske men ikke nødvendigvis ekvivalente
- ekvivalente og ikke identiske
- ekvivalente men ikke nødvendigvis identiske
- ekvivalente og identiske

18 INFO132: Flervalg - type

type(float) gir resultat
Velg ett alternativ:

- 'float'
- float
- <class 'type'>
- <class 'builtin_function_or_method'>

19 INFO132: Flervalg - type

type(lambda p, r: 2*p*r / (p + r)) gir resultat
Velg ett alternativ:

- <class 'function'>
- <class 'str'>
- <class 'builtin_function_or_method'>
- <class 'type'>

20 INFO132: Flervalg - intervall

```
for i in range(1, 6, -1):
    print(i)
```

gir som utskrift
Velg ett alternativ:

- 5 4 3 2 1
- Ingenting - utskriften er tom
- 6 5 4 3 2
- 1 0 -1 -2 -3 -4

21 INFO132: Flervalg - fortegnelse

{ 'Kina': 2, 'Indonesia': 4, 'India': 1, 'USA': 3}[3] gir resultat
Velg ett alternativ:

- Key error: '3'
- ('USA': 3)
- 3
- 'USA'

22 INFO132: Flervalg - mengder

{'A', 'b', 'r', 'a'} | {'k', 'a', 'd', 'a', 'b', 'r', 'a'} kan gi som resultat
Velg ett alternativ:

- {'A', 'b', 'r', 'a', 'k', 'a', 'd', 'a', 'b', 'r', 'a'}
- {'A', 'b', 'r', 'd'}
- {'a', 'k', 'd', 'A', 'r', 'b'}
- True

23 INFO132: Flervalg - fortegnelse

{ 'Kina': 2, 'Indonesia': 4, 'India': 1, 'USA': 3}['USA']
Velg ett alternativ:

- ('USA', 3)
- Key error: 'USA'
- {'USA': 3}
- 3

24 INFO132: Flervalg - fortegnelse

list({ 'Kina': 2, 'Indonesia': 4, 'India': 1, 'USA': 3 }.items())[1] gir resultat
Velg ett alternativ:

- Key error: 1
- ('India', 1)
- 'India'
- ('Indonesia', 4)

25 INFO132: Flervalg - listebygger

[i // 3 for i in range(4, 16, 4)] gir resultat
Velg ett alternativ:

- [x] [1, 2, 4]
- [ ] [1, 2, 0]
- [ ] [1, 2, 4, 5]
- [ ] [1, 2, 0, 1]

26 INFO132: Flervalg - listebygger

[[f'{b}{i}' for i in range(1, 4)] for b in 'ABC'] gir resultat
Velg ett alternativ:

- [ ] [['A1', 'B1', 'C1'], ['A2', 'B2', 'C2'], ['A3', 'B3', 'C3']]
- [x] [['A1', 'A2', 'A3'], ['B1', 'B2', 'B3'], ['C1', 'C2', 'C3']]
- [ ] ['ABC1', 'ABC2', 'ABC3']
- [ ] ['A1', 'B1', 'C1', 'A2', 'B2', 'C2', 'A3', 'B3', 'C3']

27 INFO132: Flervalg - formattering

nedbør = 9.87654321
id = 1234
Nå gir uttrykkene
f'Målestasjon {id:06d}: {nedbør:2.2f}ml nedbør'
og
'Målestasjon %06d: %2.2fml nedbør' % (id, nedbør)
samme tegnstreng
Velg ett alternativ:

- 'Målestasjon 1234: 9.9ml nedbør'
- 'Målestasjon 001234: 9.87ml nedbør'
- 'Målestasjon 1234: 9.88ml nedbør'
- 'Målestasjon 001234: 9.88ml nedbør'

28 INFO132: Flervalg - parameter

def registrer(*deltagere):
Velg ett alternativ:

- parameternavnet er '*deltagere' og inneholder en stjerne
- deltagere er en opsjon med [] som standardverdi
- deltagere er en sekvens av ukjent lengde
- deltagere er en opsjon med None som standardverdi

29 INFO132: Flervalg - GUI

Hvilken er *ikke* et GUI-element (widget) i **tkinter**?
Velg ett alternativ:

- Button
- Mouse
- TopLevel
- Frame

30 INFO132: Flervalg - setninger og uttrykk

Hva slags programmeringsteknikk er brukt her:

```
if id in målinger and målinger[id] >= 10.0:
    ...
```

Velg ett alternativ:

- Betinget uttrykk ("conditional expression")
- Betinget alternativ utførelse ("chained conditional")
- Unntak
- Avkortet evaluering ("short-circuit evaluation")

31 INFO132: Flervalg - funksjoner

I programlinjen
def lagre_målinger(filnavn, område=None):
er **None**
Velg ett alternativ:

- en parameter
- en standardverdi
- en opsjon
- et argument

32 INFO132: Flervalg - funksjoner

I programlinjen
def lagre_målinger(filnavn, område=None):
er **filnavn**
Velg ett alternativ:

- en standardverdi
- en opsjon
- et argument
- en parameter

33 INFO132: Flervalg - reserverte ord

Hvilket er *ikke* et reservert ord i Python?
Velg ett alternativ:

- [x] self
- [ ] class
- [ ] del
- [ ] return

34 INFO132: Flervalg - inndata

Hvilken type returneres av programsetningen **input(f'Tast inn nedbør for målestasjon {id}: ')**
Velg ett alternativ:

- [ ] float
- [ ] Det kommer an på hva brukeren taster inn
- [ ] int
- [x] str

35 INFO132: Flervalg - navnerom

I Python hjelper *navnerom* og *skop* oss først og fremst med å holde orden på
Velg ett alternativ:

- semantiske feil
- variabelverdier
- syntaksfeil
- hvor vi er i programmet

36 INFO132: Flervalg - innebygd funksjon

Hva er *ikke* en innebygget funksjon i Python (altså i biblioteket **__builtins__**)?
Velg ett alternativ:

- input
- rename
- open
- help

37 INFO132: Flervalg - setninger og uttrykk

Hva slags programsetning eller -uttrykk er brukt her:

```
if nedbør < 0:
    print('Nedbør kan ikke være et negativt tall.')
elif nedbør > 1000:
    print('Feilaktig nedbørsmengde.')
else:
    print('Nedbørsmengden er ok.')
```

Velg ett alternativ:

- Betinget uttrykk ("conditional expression")
- Unntak
- Avkortet evaluering ("short-circuit evaluation")
- Betinget alternativ utførelse ("chained conditional")

38 INFO132: Flervalg - setninger og uttrykk

Hva slags programsetning eller -uttrykk er brukt her:

```
x-y if x>y else 0
```

Velg ett alternativ:

- Unntak
- Avkortet evaluering ("short-circuit evaluation")
- Betinget uttrykk ("conditional expression")
- Betinget alternativ utførelse ("chained conditional")

39 INFO132: Flervalg - presedens

Hva er presedensrekkefølgen i Python?
Velg ett alternativ:

- and før or før not
- not før and før or
- or før and før not
- not før or før and

40 INFO132: Flervalg - presedens

Hvordan blir uttrykket utført?
11*4+23+8/4*2-9**
Velg ett alternativ:

- (11*4)+(((2**3)+((8/4)*2))-9)
- (11*(4+2))**(((3+8)/4)*(2-9))
- ((11*4)+((2**3)+((8/4)*2)))-9
- (11*(4+2))**((3+8)/(4*(2-9))

41 INFO132: Flervalg - presedens

Hva er riktig presedensrekkefølge? (PEMDAS er forklart i læreboken og på forelesning.)
Velg ett alternativ:

- PEMDAS før sammenligninger før logiske operatorer
- PEMDAS før logiske operatorer før sammenligninger
- logiske operatorer før sammenligninger før PEMDAS
- logiske operatorer før PEMDAS før sammenligninger

42 INFO132: Flervalg - overbelastning

Overbelastning i Python betyr at
Velg ett alternativ:

- En funksjon, metode eller operator har ulik betydning i ulike sammenhenger
- Samme variabel er brukt til forskjellige ting ulike steder i programmet
- Programkoden krever for store maskinressurser
- En variabel tilordnes flere ulike verdier samtidig

43 INFO132: Flervalg - print

I programsetningen **print('Hallo', 'INFO132', end='!')** er **print()**
Velg ett alternativ:

- en metode
- en funksjon
- et uttrykk
- en kommando

44 INFO132: Flervalg - break

break i en Python-løkke
Velg ett alternativ:

- Avsluttter runden og løkken med det samme
- Gjør runden ferdig og fortsetter løkken
- Gjør runden ferdig og avsluttter løkken
- Avsluttter runden med det samme men fortsetter løkken

45 INFO132: Flervalg - continue

continue i en Python-løkke
Velg ett alternativ:

- (•) Avsluttter runden med det samme men fortsetter løkken
- ( ) Avsluttter runden med det samme og går ut av løkken
- ( ) Gjør runden ferdig og fortsetter løkken
- ( ) Gjør runden ferdig og avsluttter løkken

46 INFO132: Flervalg - global

I Python betyr **global x** at
Velg ett alternativ:

- (•) vi skal bruke x i nærmeste omsluttende skop
- ( ) variabelnavnet x bare kan brukes i hovedprogrammet (__main__)
- ( ) vi skal bruke x i det globale skopet
- ( ) x er ikke kan forandre verdi (immutabilitet)

47 INFO132: Flervalg - mengder

En Python-mengde (**set**) er
Velg ett alternativ:

- en forandrbar sekvenstype
- uforandrbar og ikke en sekvenstype
- en uforandrbar sekvenstype
- forandrbar, men ikke en sekvenstype

48 INFO132: Flervalg - GUI

For å lese inn en tegnstreng fra brukeren kan vi instansiere **tkinter**-klassen
Velg ett alternativ:

- StringVar
- Entry
- Label
- Input

49 INFO132: Flervalg - GUI

Tre sentrale begrep i hendelsesdrevet programmering ("event-driven programming") er:
Velg ett alternativ:

- Modul, navnerom ("name space") og skop
- Vindu, ramme ("frame") og element
- Element, hendelse ("event") og håndterer ("handler")
- Objekt, variabel og verdi

50 INFO132: Flervalg - variabler

En Python-variabel kan *ikke* inneholde
Velg ett alternativ:

- En verdi
- Den kan inneholde alle disse
- En funksjon
- Et objekt

51 INFO132: Flervalg - klasser

Det mest presise er å si at en Python-klasse kan definere
Velg ett alternativ:

- Funksjoner og variable
- Metoder og variable
- Funksjoner og attributter
- Metoder og attributter

52 INFO132: Flervalg - GUI

tkinter-metodene **grid()** og **pack()**
Velg ett alternativ:

- Brukes kun hvis man vil kontrollere plasseringen av GUI-elementer
- pack() brukes for å lage GUI-er for små skjermer
- Uten en av dem vises ikke GUI-elementet
- grid() brukes for å presentere og editere data som tabeller

53 INFO132: Flervalg - filer

fil.close()
Velg ett alternativ:

- Kun nødvendig for read-modus
- Nødvendig for både read-, write- og append-modus
- Kun nødvendig for write- og append-modus
- Kun tillatt for append- og write-modus

54 INFO132: Flervalg - filer

open(filnavn, 'w')
Velg ett alternativ:

- Hvis filen finnes blir nye data lagt til på slutten
- Hvis filen finnes gjøres unntak
- Hvis filen ikke finnes blir den opprettet
- Hvis filen ikke finnes gjøres unntak

55 INFO132: Flervalg - filer

open(filnavn) eller **open(filnavn, 'r')**
Velg ett alternativ:

- Hvis filen ikke finnes blir den opprettet
- Hvis filen finnes blir den overskrevet
- Hvis filen ikke finnes gjøres unntak
- Hvis filen finnes gjøres unntak

56 INFO132: Flervalg - while

while *betingelse* **:** *kropp*
Velg ett alternativ:

- Når betingelse evalueres til False avsluttes løkken med det samme
- Når betingelse evalueres til True avsluttes løkken med det samme
- Når betingelse evalueres til False utføres kropp og løkken avsluttes
- Når betingelse evalueres til True utføres kropp og løkken avsluttes

57 INFO132: Flervalg - parametre

```
def utvid_liste(liste, element):
    liste = liste + [element]

liste = [ 'India', 'Kina' ]
utvid_liste(liste, 'USA')
```

Hva skrives nå ut av **print(liste)** ?
Velg ett alternativ:

- None
- ['India', 'Kina']
- ['India', 'Kina', 'USA']
- TypeError: ...

58 INFO132: Flervalg - bygging

{b for b in 'Abrakadabra'} kan gi resultatet

Velg ett alternativ:

- {'A', 'b', 'r', 'a', 'k', 'd', 'a', 'b', 'r', 'a'}
- {0: 'Abrakadabra'}
- {'d', 'r', 'b', 'A', 'k', 'a'}
- {'Abrakadabra'}

59 INFO132: Flervalg - bygging

liste = ['brød', 'melk', 'smør']
{ liste[i]: i for i in range(len(liste)) } kan gi resultatet

Velg ett alternativ:

- {0: 'brød', 1: 'melk', 2: 'smør'}
- ['brød', 'melk', 'smør']
- {'brød', 'melk', 'smør'}
- {'brød': 0, 'melk': 1, 'smør': 2}

60 INFO132: Flervalg - fortegnelse

dict(['brød', 'melk', 'smør']) kan gi resultatet

Velg ett alternativ:

- {0: 'brød', 1: 'melk', 2: 'smør'}
- {'brød', 'melk', 'smør'}
- ValueError: ...
- {'brød': 0, 'melk': 1, 'smør': 2}

61

INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
i = 0
tabell = {}
for bokstav in 'ABC':
    for tall in range(1, 4):
        tabell[(bokstav, tall)] = i
        i += 1
print(tabell)
```

Velg ett alternativ:

- TypeError: list indices must be integers or slices, not tuple
- {('A', 1): 0, ('A', 2): 1, ('A', 3): 2, ('B', 1): 3, ('B', 2): 4, ('B', 3): 5, ('C', 1): 6, ('C', 2): 7, ('C', 3): 8}
- {'A': {1: 0, 2: 1, 3: 2}, 'B': {1: 3, 2: 4, 3: 5}, 'C': {1: 6, 2: 7, 3: 8}}
- [[0, 1, 2], [3, 4, 5], [6, 7, 8]]

62

INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
dverger = [
    'Dainn', 'Dvalinn', 'Nyi', 'Nidi', 'Nordri', 'Sudri', 'Austri', 'Vestri'
]

for i in range(len(dverger)-1, 3, -1):
    print(dverger[i], end=' ')
```

Velg ett alternativ:

- Vestri Austri Sudri Nordri
- Nordri Sudri Austri
- Vestri Austri Sudri
- Nordri Sudri Austri Vestri

63 # INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
dverger = [
    'Dainn', 'Dvalinn', 'Nyi', 'Nidi', 'Nordri', 'Sudri', 'Austri', 'Vestri'
]

for i in range(len(dverger)):
    if i % 3 == 0:
        print(dverger[i], end=' ')
```

Velg ett alternativ:

- Dvalinn Nordri Vestri
- Dainn Dvalinn Nyi
- Dainn Nidi Austri
- Dainn Dainn Dainn

64 INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
a = 'Opprinnelig verdi'
b = 'Opprinnelig verdi'

def endre_variable():
    global a
    a, b = 'Ny verdi', 'Ny verdi'

endre_variable()
print('a =', a, 'og b =', b)
```

Velg ett alternativ:

- a = Opprinnelig verdi og b = Ny verdi
- a = Ny verdi og b = Opprinnelig verdi
- a = Opprinnelig verdi og b = Opprinnelig verdi
- a = Ny verdi og b = Ny verdi

65

INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
a = ['elem1', 'elem2', 'elem3']
b = ['elem1', 'elem2', 'elem3']

def endre_variable():
    global a
    a[1] = 'nytt elem2'
    b[2] = 'nytt elem3'

endre_variable()
print('a =', a, 'og b =', b)
```

Velg ett alternativ:

- a = ['elem1', 'nytt elem2', 'elem3'] og b = ['elem1', 'elem2', 'elem3']
- a = ['elem1', 'nytt elem2', 'nytt elem3'] og b = ['elem1', 'nytt elem2', 'nytt elem3']
- a = ['elem1', 'nytt elem2', 'elem3'] og b = ['elem1', 'elem2', 'nytt elem3']
- a = ['elem1', 'elem2', 'elem3'] og b = ['elem1', 'elem2', 'elem3']

66

INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
from copy import deepcopy

matrise = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

def endre_matrise(matrise):
    matrise[1][1] = 100

kopi = matrise.copy()
dyp_kopi = deepcopy(matrise)

endre_matrise(matrise)
endre_matrise(kopi)
endre_matrise(dyp_kopi)

print(f'matrise[1][1]={matrise[1][1]}, '
      f'kopi[1][1]={kopi[1][1]} og '
      f'dyp_kopi[1][1]={dyp_kopi[1][1]}')
```

Velg ett alternativ:

- matrise[1][1]=5, kopi[1][1]=5 og dyp_kopi[1][1]=100
- matrise[1][1]=100, kopi[1][1]=100 og dyp_kopi[1][1]=100
- matrise[1][1]=5, kopi[1][1]=5 og dyp_kopi[1][1]=5
- matrise[1][1]=5, kopi[1][1]=100 og dyp_kopi[1][1]=100

67

INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
class Superklasse:
    def __init__(self, navn):
        self.navn = navn
        self.beskrivelse = 'Jeg tilhører superklassen'

class Subklasse(Superklasse):
    def __init__(self, navn, verdi):
        self.verdi = verdi
        self.beskrivelse = 'Jeg tilhører subklassen'
        super().__init__(navn)

objekt1 = Superklasse('Superklasseobjekt')
objekt2 = Subklasse('Subklasseobjekt', 25)

print(f'{objekt1.navn}: {objekt1.beskrivelse}')
print(f'{objekt2.navn}, {objekt2.verdi}: {objekt2.beskrivelse}')
```

Velg ett alternativ:

- Superklasseobjekt: Jeg tilhører superklassen
  Subklasseobjekt, 25: Jeg tilhører subklassen
- Superklasseobjekt: Jeg tilhører subklassen
  Subklasseobjekt, 25: Jeg tilhører superklassen
- Superklasseobjekt: Jeg tilhører subklassen
  Subklasseobjekt, 25: Jeg tilhører subklassen
- Superklasseobjekt: Jeg tilhører superklassen
  Subklasseobjekt, 25: Jeg tilhører superklassen

68 # INFO132: Programforståelse

Reglene for små og store bokstaver i engelske titler er annerledes enn de norske.

Hva skriver dette Python-programmet ut?

```
def engelsk_tittel(tekst):
    # lister over ord som skal ha stor eller liten forbokstav
    artikler = ["a", "an", "the"]
    preposisjoner = ["in", "on", "over", "with", "at", "from", "into", "during", "including",
"until",
                     "against", "among", "throughout", "despite", "towards", "upon",
"concerning"]
    konjunksjoner = ["and", "but", "or", "nor", "for", "so", "yet"]
    ord = tekst.split()
    nye_ord = []
    for i, ord in enumerate(ord):
        if i == 0 or i == len(ord) - 1:
            nye_ord.append(ord.capitalize())
        elif ord.lower() not in artikler + preposisjoner + konjunksjoner:
            nye_ord.append(ord.capitalize())
        else:
            nye_ord.append(ord.lower())
    return " ".join(nye_ord)

# eksempel på kjøring
tittel = "the QuIcK brown FOX jumps over The lazy doG"
ny_tittel = engelsk_tittel(tittel)
print(ny_tittel)
```

Velg ett alternativ:

- The Quick Brown Fox Jumps over The Lazy Dog
- The QuIcK Brown FOX Jumps over The Lazy DoG
- The QuIcK Brown FOX Jumps over the Lazy DoG
- The Quick Brown Fox Jumps over the Lazy Dog

69 INFO132: Programforståelse

Hanois tåren ("Tower of Hanoi") går ut på å flytte en serie med skiver fra én pinne til en annen, med hjelp av en midlertidig pinne, samtidig som man følger et sett med regler: bare én skive kan flyttes om gangen, og ingen skive kan plasseres oppå en mindre skive.

Hva skriver dette Python-programmet ut?

```
def tower_of_hanoi(n, fra_pinne, til_pinne, hjelpepinne):
    if n == 1:
        print(f'Skive 1 fra {fra_pinne} til {til_pinne}', end='. ')
        return
    tower_of_hanoi(n-1, fra_pinne, hjelpepinne, til_pinne)
    print(f'Skive {n} fra {fra_pinne} til {til_pinne}', end='. ')
    tower_of_hanoi(n-1, hjelpepinne, til_pinne, fra_pinne)

n = 2  # antall skiver
tower_of_hanoi(n, fra_pinne='A', til_pinne='C', hjelpepinne='B')
```

Velg ett alternativ:

- Skive 1 fra C til B. Skive 2 fra A til B. Skive 1 fra B til C.
- Skive 1 fra B til C. Skive 2 fra A til C. Skive 1 fra A til B.
- Skive 1 fra A til C. Skive 2 fra B til A. Skive 1 fra C til B.
- Skive 1 fra A til B. Skive 2 fra A til C. Skive 1 fra B til C.

70 INFO132: Programforståelse

Hva skriver dette Python-programmet ut?

```
def iterativ_binærsøk(fortegnelser, mål_id):
    start = 0
    slutt = len(fortegnelser) - 1
    while start <= slutt:
        midt = (start + slutt) // 2
        midt_id = fortegnelser[midt]['id']
        print(f'Søker etter id {mål_id} mellom posisjon {start} og {slutt} (inklusiv).')
        if midt_id == mål_id:
            return fortegnelser[midt]['navn']
        elif mål_id < midt_id:
            slutt = midt - 1
        else:
            start = midt + 1
    return 'Id ikke funnet'

# eksempel på bruk
fortegnelser = [
    {'id': id, 'navn': f'Person nr {id:03d}'}
    for id in range(1, 101, 3)
]
mål_id = 76
målperson = iterativ_binærsøk(fortegnelser, mål_id)
print('Fant:', målperson)
```

Velg ett alternativ:

Søker etter id 76 mellom posisjon 0 og 33 (inklusiv).
Søker etter id 76 mellom posisjon 0 og 15 (inklusiv).
Fant: Person nr 076

Søker etter id 76 mellom posisjon 0 og 33 (inklusiv).
Søker etter id 76 mellom posisjon 17 og 33 (inklusiv).
Søker etter id 76 mellom posisjon 17 og 24 (inklusiv).
Søker etter id 76 mellom posisjon 21 og 24 (inklusiv).
Søker etter id 76 mellom posisjon 23 og 24 (inklusiv).
Søker etter id 76 mellom posisjon 24 og 24 (inklusiv).
Fant: Id ikke funnet

Søker etter id 76 mellom posisjon 0 og 33 (inklusiv).
Søker etter id 76 mellom posisjon 17 og 33 (inklusiv).
Søker etter id 76 mellom posisjon 26 og 33 (inklusiv).
Fant: Person nr 076

Søker etter id 76 mellom posisjon 0 og 33 (inklusiv).
Søker etter id 76 mellom posisjon 17 og 33 (inklusiv).
Fant: Person nr 076

71 INFO132: Programutfylling

I en FIbonacci-sekvens (også kalt Fibonacci-følge) er hvert tall summen av de to foregående. Den enkleste og mest kjente Fibonacci-sekvensen starter med tallene 0 og 1, og fortsetter som følger:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Her er en iterativ funksjon som skriver ut slike Fibonacci-sekvenser med angitt lengde:

```
def fibonacci_iterativ(n):
    if n == 0:
        return
    elif n == 1:
        print(1)
        return
    a, b = 0, 1
    print(a, b, sep=', ', end='')
    for i in range(3, n + 1):
        [programkoden mangler her]
        print(',', b, end='')

# testkjøring
fibonacci_iterativ(10)
```

Utskriften skal bli:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- a = b, b = a
- b = a + b
- b = a
- a, b = b, a + b

72 INFO132: Programutfylling

Her er en Python-funksjon som leser linjer fra en fil. Hver linje i filen inneholder et navn og et postnummer adskilt med tabulatortegn ('\t'). Funksjonen returnerer en liste over navn som samsvarer med et gitt postnummer.

```
def finn_navn_etter_postnummer(filnavn, postnummer):
    passende_navn = []
    with open(filnavn, 'r') as fil:
        for linje in fil:
            [programkoden mangler her]
            if nummer == postnummer:
                passende_navn.append(navn)
    return passende_navn

# testkjøring
filnavn = 'data1.txt'  # erstatt med ditt filnavn
postnummer = '2345'  # erstatt med målpostnummeret
print(finn_navn_etter_postnummer(filnavn, postnummer))
```

Eksempel på innhold i filen data1.txt:
Per Person\t1234
Kari Karsdatter\t2345
Ali Olaisen\t3456

Utskriften skal bli:
['Kari Karsdatter']

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- navn, nummer = linje.strip().split('\t')
- nummer, navn = linje.split(',')
- nummer, navn = linje.split('\t')
- navn, nummer = linje.strip().split(',')

73 INFO132: Programutfylling

Her er en Python-funksjon som tar en personliste og en liste over gyldige postkoder som parametre. Personlisten består av fortegnelser ("dictionaries") som representerer personer med id, navn, gateadresse og postadresse. Postadressen består av en postkode og et poststed. Funksjonen returnerer en ny liste med alle personene som har postkode i listen over gyldige postkoder.

```
import json
def filtrer_personer_etter_postkode(personer, gyldige_postkoder):
    filtrerte_personer = [programkoden mangler her]
    return list(filtrerte_personer)

# testkjøring
personer_json = '''
[    {"id": 1, "navn": "Ola Nordmann", "gateadresse": "Bjørkelia 3",
       "postadresse": {"postkode": "1234", "poststed": "Oslo"}},
     {"id": 2, "navn": "Kari Nordmann", "gateadresse": "Granveien 5",
       "postadresse": {"postkode": "5678", "poststed": "Bergen"}},
     {"id": 3, "navn": "Per Hansen", "gateadresse": "Furulund 10",
       "postadresse": {"postkode": "1234", "poststed": "Oslo"}}
]
'''
personer = json.loads(personer_json)
gyldige_postkoder = ['1234']
print(filtrer_personer_etter_postkode(personer, gyldige_postkoder))
```

Utskriften skal bli:

```
[    {'id': 1, 'navn': 'Ola Nordmann', 'gateadresse': 'Bjørkelia 3',
       'postadresse': {'postkode': '1234', 'poststed': 'Oslo'}},
     {'id': 3, 'navn': 'Per Hansen', 'gateadresse': 'Furulund 10',
       'postadresse': {'postkode': '1234', 'poststed': 'Oslo'}}
]
```

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- filter(lambda person: person['postadresse']['postkode'] == gyldige_postkoder, personer)
- [person if person['postadresse']['postkode'] in gyldige_postkoder for person in personer]
- filter(lambda person: person['postadresse']['postkode'] in gyldige_postkoder, personer)
- [person for postkode in postkoder if person['postadresse']['postkode'] == postkode]

74 INFO132: Programutfylling

Her er en Python-funksjon som tar en liste av personnavn på formen 'Etternavn, Fornavn Mellomnavn1 Mellomnavn2 ...' og returnerer en liste på formen 'Fornavn Mellominitial1 Mellominitial2 ... Etternavn'.

```
def konverter_navn(personnavn_liste):

    def initialer(navn):
        deler = navn.split()
        return ' '.join([deler[0]] + [n[0] + '.' for n in deler[1:]])

    def konverter(navn):
        etternavn, resten = navn.split(', ')
        return f'{initialer(resten)} {etternavn}'

    return [programkoden mangler her]

# testkjøring
personnavn_liste = ['Doe, John', 'Nordmann, Ola Petter', 'Hansen, Anna Lise']
konverterte_navn = konverter_navn(personnavn_liste)
print(konverterte_navn)
```

Utskriften skal bli:
['John Doe', 'Ola P. Nordmann', 'Anna L. Hansen']

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- list(map(konverter, personnavn_liste))
- konverter(personnavn_liste)
- map(lambda navn: konverter(navn), personnavn_liste)
- list(map(lambda navn: konverter, personnavn_liste))

75 INFO132: Programutfylling

Her er et **tkinter**-program med en etikett ('label') og en styrespake ('slider'). Når man skyver på styrespaken skal skriftstørrelsen til etiketten endres.

```
import tkinter as tk
from tkinter import font

rot = tk.Tk()
rot.title('Juster skriftstørrelse')

styrespake = tk.Scale(
   rot, from_=6, to=20, orient='horizontal',
   command=[programkoden mangler her]
)
styrespake.pack()

etikett = tk.Label(rot, text='Juster skriftstørrelsen', font=('Helvetica', 6))
etikett.pack()

# start hendelsesløkken
rot.mainloop()
```

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- lambda styrespake: styrespake.get()
- etikett.config(font=('Helvetica', styrespake.get()))
- lambda _: etikett.config(font=('Helvetica', styrespake.get()))
- etikett.config(font=styrespake.get())

76 INFO132: Programutfylling

Her er en Python-funksjon som tar to parametre: et filnavn og en fortegnelse ("dictionary") med id, navn og adresse til en person. I filen er det allerede lagret en person på hver linje, med id, personnavn og adresse adskilt med tabulatortegn ('\t'). Funksjonen skal skrive personen i fortegnelsen til slutten av filen hvis den ikke allerede er lagret der.

```
def legg_til_person_hvis_ikke_finnes(ny_person, filnavn):
    # leser eksisterende data fra filen
    eksisterende_personer = []
    with open(filnavn, 'r') as file:
        for linje in file:
            persondata = linje.strip().split('\t')
            id, navn, adresse = persondata
            eksisterende_personer.append({
                'id': id,
                'navn': navn,
                'adresse': adresse
            })
    # sjekker om personen allerede finnes i filen
    if any(person['id'] == ny_person['id'] for person in eksisterende_personer):
        print(f'Personen {ny_person["navn"]} finnes allerede i filen.')
    else:
        [programkoden mangler her]
        print(f'Ny person {ny_person["navn"]} lagt til i filen.')

# testkjøring
eli_json = {'id': '123', 'navn': 'Eli Wang', 'adresse': 'Bjørkelia 3'}
ola_json = {'id': '456', 'navn': 'Ola Nordmann', 'adresse': 'Granlia 5'}
filnavn = 'personer.txt'
legg_til_person_hvis_ikke_finnes(eli_json, filnavn)
legg_til_person_hvis_ikke_finnes(ola_json, filnavn)
```

Eksempel på innhold i filen personer.txt:
456\tOla Nordmann\tGranlia 5

Utskriften skal bli:
Ny person Eli Wang lagt til i filen.
Personen Ola Nordmann finnes allerede i filen.

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- open(filnavn).append('\t'.join(ny_person.values())+'\n')
- open(filnavn, 'a').write('\t'.join(ny_person.values())+'\n')
- filnavn.append(ny_person)
- filnavn.write(ny_person + '\n', mode='a')

77 INFO132: Programutfylling

Her er en Python-funksjon som tar en liste som parameter. Listen består av fortegnelser ("dictionaries") som representerer personer med id, navn, gateadresse, postkode og kanskje poststed. Det finnes dessuten en global fortegnelse som kobler postkoder til poststeder. Hvis en personfortegnelse mangler poststed og postkoden finnes i den globale fortegnelsen så legges poststedet til personfortegnelsen. Hvis personfortegnelsen derimot har både postkode og -sted, men postkoden ikke finnes i den globale fortegnelsen, så skal den legges til der. Til slutt returneres den oppdaterte listen over personer.

```
# eksempel på global fortegnelse over postkoder og -steder
postkoder = {'1234': 'Oslo', '5600': 'Bergen'}

def oppdater_personer_og_poststeder(personer):
    oppdaterte_personer = []
    for person in personer:
        postkode = person.get('postkode')
        poststed = person.get('poststed')
        if poststed is None and postkode in postkoder:
            person['poststed'] = postkoder[postkode]
        elif [programkoden mangler her]
            postkoder[postkode] = poststed
        oppdaterte_personer.append(person)
    return oppdaterte_personer

# testkjøring
personer_json = [
    {'id': '1', 'navn': 'Ola Nordmann', 'gateadresse': 'Bjørkelia 3', 'postkode': '1234'},
    {'id': '2', 'navn': 'Kari Nordmann', 'gateadresse': 'Granveien 5', 'postkode': '5600',
'poststed': 'bergen'},
    {'id': '3', 'navn': 'Per Hansen', 'gateadresse': 'Furulund 10', 'postkode': '9999',
'poststed': 'Trondheim'}
]
print('Oppdaterte personer:', oppdater_personer_og_poststeder(personer_json))
print('Nye postkoder:', postkoder)
```

Utskriften skal bli noe sånt som:

```
Oppdaterte personer: [
    {'id': '1', 'navn': 'Ola Nordmann', 'gateadresse': 'Bjørkelia 3', 'postkode': '1234',
'poststed': 'Oslo'},
    {'id': '2', 'navn': 'Kari Nordmann', 'gateadresse': 'Granveien 5', 'postkode': '5600',
'poststed': 'Bergen'},
    {'id': '3', 'navn': 'Per Hansen', 'gateadresse': 'Furulund 10', 'postkode': '9999',
'poststed': 'Trondheim'} ]
Nye postkoder: {'1234': 'Oslo', '5600': 'Bergen', '9999': 'Trondheim'}
```

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- poststed is not None:
- not (postkode in postkoder or poststed is not None):
- postkode not in postkoder and poststed is not None:
- postkode not in postkoder or poststed is not None:

78 INFO132: Programutfylling

Her er en Python-funksjon som tar en liste av fortegnelser ("dictionaries") som parameter. Hver fortegnelse inneholder en id og et personnavn. Listen er sortert etter id. Funksjonen utnytter at listen er sortert til å finne navnet som hører til en oppgitt id. Underveis skriver funksjonen også ut informasjon om søkeprosessen (se eksemplet under).

```
def binærsøk(fortegnelser, mål_id, start, slutt):
    if start > slutt:
        return 'Id ikke funnet'
    midt = (start + slutt) // 2
    midt_id, midt_navn = fortegnelser[midt]['id'], fortegnelser[midt]['navn']
    print(f'Søker etter id {mål_id} mellom posisjon {start} og {slutt} (inklusiv).')
    if midt_id == mål_id:
        return midt_navn
    elif mål_id < midt_id:
        return [programkode (a) mangler her]
    else:
        return [programkode (b) mangler her]

# testkjøring
fortegnelser = [
    {'id': id, 'navn': f'Person nr {id:03d}'}
    for id in range(1, 101, 3)
]
mål_id = 22
målperson = binærsøk(fortegnelser, mål_id, 0, len(fortegnelser) - 1)
print('Fant:', målperson)
```

Utskriften skal bli:

```
Søker etter id 22 mellom posisjon 0 og 33 (inklusiv).
Søker etter id 22 mellom posisjon 0 og 15 (inklusiv).
Fant: Person nr 022
```

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

a) binærsøk(fortegnelser, mål_id, start, midt-1)
b) binærsøk(fortegnelser, mål_id, slutt, midt+1)

a) binærsøk(fortegnelser, mål_id, start+1, midt-1)
b) binærsøk(fortegnelser, mål_id, slutt-1, midt+1)

a) binærsøk(fortegnelser, mål_id, start, midt-1)
b) binærsøk(fortegnelser, mål_id, midt+1, slutt)

a) binærsøk(fortegnelser, mål_id, start, midt)
b) binærsøk(fortegnelser, mål_id, midt, slutt)

79 INFO132: Programutfylling

Her er en Python-funksjon som holder orden på en (global) fortegnelse ("dictionary") over ansatte og hva de kan (altså deres kompetanser). Parametrene til funksjonen er et personnavn og en liste av kompetanser - hver representert som en tegnstreng. Hvis personen allerede er registrert i fortegnelsen blir personens kompetanser oppdatert. Hvis personen derimot ikke er registrert fra før så legges personen til med sine kompetanser. Merk at eksisterende kompetanser aldri skal slettes og ingen kompetanser skal registreres to ganger for samme person.

```
# eksempel på global fortegnelse
ansattkompetanser = {
    'Per Person': ['Python-programmering', 'Maskinlæring'],
    'Dom Dumrian': []
}

def legg_til_kompetanse(personnavn, kompetanser):
    if personnavn in ansattkompetanser:
        [programkode (a) mangler her]
    else:
        [programkode (b) mangler her]

# testkjøring
legg_til_kompetanse('Ola Nordmann', ['Python-programmering'])
legg_til_kompetanse('Kari Nordmann', ['Java-programmering', 'Systemdesign'])
legg_til_kompetanse('Ola Nordmann', ['Dataanalyse', 'Python-programmering'])
print(ansattkompetanser)
```

Utskriften skal bli noe sånt som:

```
{   'Per Person': ['Python-programmering', 'Maskinlæring'],
    'Dom Dumrian': [],
    'Ola Nordmann': {'Dataanalyse', 'Python-programmering'},
    'Kari Nordmann': {'Systemdesign', 'Java-programmering'}   }
```

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

a) ansattkompetanser[personnavn].add(kompetanser)
b) ansattkompetanser[personnavn] = set(kompetanser)

a) ansattkompetanser[personnavn].update(kompetanser)
b) ansattkompetanser[personnavn] = set(kompetanser)

a) ansattkompetanser[personnavn].extend(kompetanser)
b) ansattkompetanser[personnavn] = kompetanser[:]

a) ansattkompetanser[personnavn].extend(kompetanser)
b) ansattkompetanser[personnavn] = set(kompetanser)

80 INFO132: Programutfylling

Her er en Python-funksjon som tar en person-id som parameter. Hvis det allerede finnes en fil med navnet '<person_id>.json' returnerer funksjonen en fortegnelse ("dictionary") med informasjon som leses fra filen. Hvis en slik fil ikke finnes hentes personinformasjon i stedet fra vevadressen 'http://eksamen.info132.uib.no/personinfo/<person_id>' (en URL) ved hjelp av Python's requests API. Informasjonen som hentes fra vev-API-et ('web API') er i samme JSON-format som brukes i filene. Informasjonen lagres så i en ny JSON-fil med navnet <person_id>.json før JSON-objektet returneres.

```
import json
import requests
import os

vevapi = 'https://api.semanticscholar.org/graph/v1/author/'

def hent_personinfo(person_id):
    filnavn = f'{person_id}.json'

    # eksisterer filen lokalt?
    if [programkode mangler her]
        with open(filnavn, 'r') as fil:
            print('Fant lokal fil.')
            return json.load(fil)

    # ellers hentes data fra nettadressen
    url = f'{vevapi}{person_id}'
    response = requests.get(url)

    # var forespørselen vellykket?
    if response.status_code == 200:
        print('Hentet data fra vev-API.')
        person_data = response.json()
        # lagre dataene i en ny fil
        with open(filnavn, 'w') as fil:
            json.dump(person_data, fil, indent=4)
        return person_data
    else:
        print('Kunne ikke hente data fra vev-API.')
        return None

# testkjøring
person_id = '12345'
personinfo = hent_personinfo(person_id)
print(personinfo)
personinfo = hent_personinfo(person_id)
```

Utskriften skal bli noe sånt som:

```
Hentet data fra vev-API.
{'authorId': '12345', 'name': 'A. Nonym'}
Fant lokal fil.
```

Hvilken kodebit må settes inn for at programmet kjører som det skal?

Velg ett alternativ:

- os.path.isfile(filnavn)
- os.path.isdir(filnavn):
- os.path.exists(person_id)
- os.path.isfile(response)

81 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Her er begynnelsen på klassen Student:

```
class Student:
    def __init__(self, navn, epost, telefonnummer, språk='norsk'):
        self.navn = navn
        self.epost = epost
        self.tlfnr = telefonnummer
        self.språk = språk

    def skriv_ut(self):
        print(self.til_streng())

    def til_streng(self):
        # ikke ferdig
        pass
```

Gjør ferdig et kodeskjelett for resten av klassen Student, med følgende metoder **nytt_utlån()** og **lever_tilbake()**. Du skal bare skrive metodehodene (altså **def**-linjene) og ikke metodekroppene. Bruk gjerne **pass**-instruksjonen i stedet for metodekropp.

Skriv ditt svar her

```
1  class Student:
2      def __init__(self, navn, epost, telefonnummer, språk='norsk'):
3          self.navn = navn
4          self.epost = epost
5          self.tlfnr = telefonnummer
6          self.språk = språk
7
8      def skriv_ut(self):
9          print(self.til_streng())
10
11     def til_streng(self):
12         # ikke ferdig
13         pass
14
15     def nytt_utlån(self):
16         pass
17
18     def lever_tilbake(self):
19         pass
```

82 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Gjør ferdig metoden **til_streng()** i **Student**-klassen slik at

>>> arne = Student('Arne Andersen', 'aa345@student.uib.no', '97575375')
>>> arne.skriv_ut()

gir utskriften

Arne Andersen <aa345@student.uib.no>, tlf: 97575375, språk: norsk

Skriv ditt svar her

```
1  class Student:
2      def __init__(self, navn, epost, telefonnummer, språk='norsk'):
3          self.navn = navn
4          self.epost = epost
5          self.tlfnr = telefonnummer
6          self.språk = språk
7
8      def skriv_ut(self):
9          print(self.til_streng())
10
11     def til_streng(self):
12         return f"{self.navn} <{self.epost}>, tlf: {self.telefonnummer}, språk: {sel
13
14     def nytt_utlån(self):
15         pass
16
17     def lever_tilbake(self):
18         pass
```

83 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Skriv klassen **Utstyr** i henhold til følgende spesifikasjoner:

- Hvert utstyrsobjekt skal ha en **type** og en unik identifikator (**id**), samt en valgfri **størrelse** med **None** som standardverdi. Ellers lagres alle objektattributter som tegnstrenger.
- Klassen skal definere disse metodene: konstruktør (**__init__**), **til_streng()** og **skriv_ut()**. **skriv_ut()**-metoden skal kalle **til_streng()**.

Eksempler på kjøring:

>>> klatresele = Utstyr('klatresele', '264264', 'M')
>>> klatresele.skriv_ut()
Klatresele (264264), størrelse: M
>>> kalkpose = Utstyr('kalkpose', '353424')
>>> kalkpose.skriv_ut()
Kalkpose (353424)

Skriv ditt svar her

```
class Utstyr:
    def __init__(self, utstyr, nr, størrelse="None"):
        self.utstyr = utstyr
        self.nr = nr
        self.størrelse = størrelse

    def skriv_ut(self):
        print(self.til_streng())

    def til_streng(self):
        return f"{self.utstyr.capitalize()} {self.nr}, størrelse: {self.størrelse}"
```

84 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Klassen **Utstyr** skal også ha en **utstyrsliste** som klasseatributt. Utstyrslisten skal inneholde alle utstyrsobjektene som er opprettet. Utvid klassen for å få til dette, og lag en ny **list_utstyr()**-metode for klassen som skriver ut alt utstyret.

Eksempler på kjøring:

>>> klatresele = Utstyr('klatresele', '264264', 'M')
>>> kalkpose = Utstyr('kalkpose', '353424')
>>> Utstyr.list_utstyr()
Kalkpose (353424)
Klatresele (264264), størrelse: M

Skriv ditt svar her

```
1  class Utstyr:
2      def __init__(self, utstyr, nr, størrelse=None)
3          self.utstyr = utstyr
4          self.nr = nr
5          self.størrelse = størrelse
6
7      def skriv_ut(self):
8          print(self.til_streng())
9
10     def til_streng(self):
11         return f"{self.utstyr.capitalize()} {self.nr}, størrelse: {self.størrelse}"
12
13     def list_utstyr(self):
14         utstyr_liste = []
15         nytt_utstyr = f"{self.utstyr.capitalize()} {self.nr}, størrelse: {self.stør
16         utstyrs_liste.append(nytt_utstyr)
17         for utstyr in utstyr_liste:
18             print(utstyr)
19
```

85 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Konstruktørmetoden ("constructor") **__init__()** i klassen **Utstyr** skal sjekke at oppgitt utstyrsidentifikator ikke allerede er i bruk. Hvis utstyrsidentifikatoren allerede er brukt skal det gis en feilmelding og nytt utstyrsobjekt skal ikke opprettes.

Eksempler på kjøring:

>>> klatresko = Utstyr('klatresko', '386693', '39')
>>> klatretau = Utstyr('klatretau', '386693', '20 meter')
Utstyrsid 386693 er allerede i bruk

>>> Utstyr.list_utstyr()
Klatresko (386693), størrelse: 39

Skriv ditt svar her

```
1  class Utstyr:
2      def __init__(self, utstyr, nr, størrelse=None):
3          self.utstyr = utstyr
4          self.størrelse = størrelse
5          self.nr = nr
6          if nr in self.list_utstyr():
7              self.utstyr_liste.pop()
8              print(f"Utstyrsid {nr} er allerede i bruk")
9
10     def skriv_ut(self):
11         print(self.til_streng())
12
13     def til_streng(self):
14         return f"{self.utstyr.capitalize()} {self.nr}, størrelse: {self.størrelse}"
15
16     def list_utstyr(self):
17         utstyr_liste = []
18         nytt_utstyr = f"{self.utstyr.capitalize()} {self.nr}, størrelse: {self.stø
19         utstyrs_liste.append(nytt_utstyr)
20         for utstyr in utstyr_liste:
21             print(utstyr)
```

86 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Skriv koden for klassen **Utlån**. Hvert utlån er gjort til en **student** og gjelder et **utstyrsobjekt**. Utlånet har også en **dato**. Utlånklassen skal ha en konstruktør (**__init__()**), en **til_streng()**-, og en **skriv_ut()**-metode.

Eksempel på kjøring:

>>> arne = Student('Arne Andersen', 'aa345@student.uib.no', '97575375')
>>> klatresele = Utstyr('klatresele', '264264', 'M')
>>> arne_låner_klatresele = Utlån(arne, klatresele, '2023-12-19')
>>> arne_låner_klatresele.skriv_ut()

gir utskriften

Utlån 2023-12-19: Arne Andersen <aa345@student.uib.no> har lånt Klatresele (264264), størrelse: M

Skriv ditt svar her

```
1  class Utlån:
2      def __init__(self, navn, utstyr, dato):
3          self.navn = student.navn
4          self.utstyr = utstyr.utstyr
5          self.dato = dato
6
7      def til_streng(self):
8          return f"Utlån {dato}: {student.navn} {student.epost} har lånt {utstyr.utst
               .størrelse})"
9
10     def skriv_ut(self):
11         print(self.til_streng())
```

87 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Skriv ferdig metoden **nytt_utlån()** i klassen **Student**. Metoden tar et **utstyrsobjekt** og en **dato** som parameter og lager et nytt utlånsobjekt. Alle utlånsobjekter for en student legges i en mengde ("set") **alle_lån**. Vis også hvordan du endrer hvordan **Student**-objekter initialiseres (**__init__()**) for å få dette til.

Oppdater dessuten metoden **til_streng()** i **Student**-klassen slik at

```
>>> dagny = Student('Dagny Danielsen', 'dd739@student.uib.no', '41646423')
>>> dagny.nytt_utlån('353424')  # id til kalkposen
>>> dagny.nytt_utlån('386693')  # id til klatreskoene
>>> dagny.skriv_ut()
```

gir utskriften

```
Dagny Danielsen <dd739@student.uib.no>, tlf: 41646423, språk: norsk.
Utlån:
    Kalkpose (353424)
    Klatresko (386693), størrelse: 39
```

Skriv ditt svar her

```
1  class Student:
2      def __init__(self, navn, epost, telefonnummer, språk='norsk'):
3          self.navn = navn
4          self.epost = epost
5          self.tlfnr = telefonnummer
6          self.språk = språk
7
8      def skriv_ut(self):
9          print(self.til_streng())
10
11     def til_streng(self):
12         return f"{self.navn} <{self.epost}>, tlf: {self.telefonnummer}, språk: {sel
               ({utstyr.nr}, størrelse{ustyr.størrelse})"
13
14     def nytt_utlån(self, utstyrsobjekt, dato):
15         pass
16
17     def lever_tilbake(self):
18         pass
```

88 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Skriv ferdig metoden **lever_tilbake()** i klassen **Student**. Hvis studenten ikke har lånt utstyret som forsøkes levert tilbake, skal det skrives ut en feilmelding.

>>> dagny.lever_tilbake('264264')
>>> dagny.lever_tilbake('353424')
>>> dagny.lever_tilbake('353424')
Dagny Danielsen <dd739@student.uib.no> har ikke lånt Kalkpose (353424)

Skriv ditt svar her

```
class Student:
    def __init__(self, navn, epost, telefonnummer, språk='norsk'):
        self.navn = navn
        self.epost = epost
        self.tlfnr = telefonnummer
        self.språk = språk

    def skriv_ut(self):
        print(self.til_streng())

    def til_streng(self):
        return f"{self.navn} <{self.epost}>, tlf: {self.telefonnummer}, språk: {sel
            ({utstyr.nr}, størrelse{ustyr.størrelse})"

    def nytt_utlån(self, utstyrsobjekt, dato):
        pass

    def lever_tilbake(self):
        if utstyr.nr not in self.utstyrsobjekt:
            print(f"{self.navn} {self.epost} har ikke lånt {utstyrsobjekt} ({utstyr
```

89 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Skriv **til_json()**-metoder i klassene **Student** og **Utlån** slik at

```
>>> jasmin = Student('Jasmin Jafari', 'jj567@student.hpv.no', '96464192', språk='engelsk')
>>> jasmin.nytt_utlån(klatresele)
>>> jasmin.nytt_utlån(klatresko)
>>> json_obj = jasmin.til_json()
```

returnerer en fortegnelse som kan konverteres til en JSON-streng

```
{
 'navn': 'Jasmin Jafari',
 'epost': 'jj567@student.hpv.no',
 'tlfnr': '96464192',
 'utlån': [
     {
      'type': 'klatresele',
      'id': '264264',
      'størrelse': 'M',
     },
     {
      'type': 'klatresko,
      'id': '386693',
      'størrelse': '39',
     }
 ]
}
```

Skriv ditt svar her

```
def til_json():
    info = {
        "navn": f"self.navn",
        "epost": f"self.epost",
        "tlfnr": f"self.telefonnummer",
        "utlån": [

            ]
    }
```

90 INFO132: Programmering

Du skal lage et system som administrerer utlån av sport- og fritidsutstyr til studenter. Systemet skal holde oversikt over tilgjengelig utstyr og holde orden på utlån og innleveringer. Systemet skal ha tre klasser: Student, Utstyr og Utlån.

Skriv en funksjon **lagre_til_fil()** som skriver alle studenter, utstyrsobjekter og utlån til en JSON-fil. Du kan gå ut fra at studentenes epostadresser er unike.

For å få dette kan du f eks:

- Skrive en **til_json()**-metode i klassen **Utstyr**.
- La klassene **Student** og **Utlån** få henholdsvis en **studentliste** og en **utlånsliste** som klasseatributter.
- Skrive klassemetoder **alle_til_json()** i alle de tre klassene.

Skriv ditt svar her

```
1
```

91 INFO132: Kommentarer til eksamen

Her kan du legge til kommentarer til eksamen, f eks om du mener du har funnet feil eller uklarheter.

Skriv ditt svar her

Ord: 0