in1000 eksamen

nye oppgavetyper dere kan møte på

forklaringsoppgave

“forklar hva som er forskjellene mellom en for-løkke og en while-løkke.”

løsningsforslag:

- for-løkker
  - går gjennom alle elementer i en samling (list, dict, range)
  - vet på forhånd hvor mange iterasjoner (repetisjoner) det blir
- while-løkker
  - er i praksis repeterte if-tester
  - vet ikke hvor mange iterasjoner det blir – må hele tiden teste
  - kan gå uendelig hvis testen aldri feiler

(flisespikking: for-løkker *kan* også gå uendelig hvis vi legger til ting i samlingen mens vi går gjennom den, men det er verken naturlig eller smart, så det *burde* ikke skje i praksis)

forbedre kode

```
tall = 3
tall2 = tall*tall
print(tall, "ganget med seg selv er", tall2)
tall = tall2
tall2 = tall*tall
print(tall, "ganget med seg selv er", tall2)
tall = tall2
tall2 = tall*tall
print(tall, "ganget med seg selv er", tall2)
tall = tall2
tall2 = tall*tall
print(tall, "ganget med seg selv er", tall2)
```

"bruk en løkke til å gjøre denne koden mer lettlest og enklere å endre senere"

Løsningsforslag:

```
tall = 3
for i in range(4):
    if i > 0:
        tall = tall2
    tall2 = tall*tall
    print(tall, "ganget med seg selv er", tall2)

# eller

tall2 = 3
for i in range(4):
    tall = tall2
    tall2 = tall*tall
    print(tall, "ganget med seg selv er", tall2)
```

“riktig verktøy”-oppgave

“i hvilke av disse tilfellene er det naturlig/lurt å bruke ordbok i stedet for liste(r)?”

- årstallene fra 2000 til 2023
- telefonliste med navn og nummer
- oversette ord fra svensk til japansk
- rekkefølge på fotball-lag i tabellen
- sjekke om studenter har godkjent obliger eller ikke

“riktig verktøy”-oppgave

“i hvilke av disse tilfellene er det naturlig/lurt å bruke ordbok i stedet for liste(r)?”

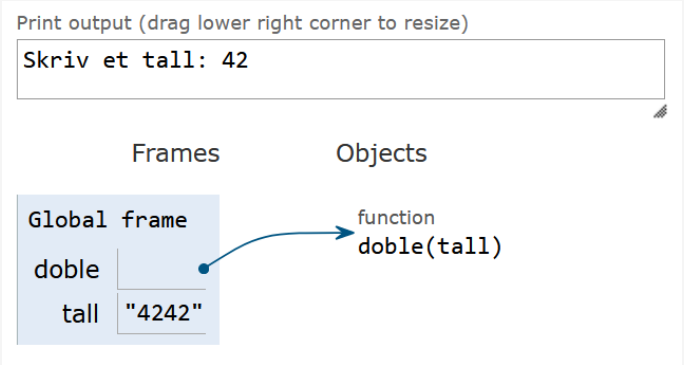
- årstallene fra 2000 til 2023
- telefonliste med navn og nummer
- oversette ord fra svensk til japansk
- rekkefølge på fotball-lag i tabellen
- sjekke om studenter har godkjent obliger eller ikke

sammenhengen vi tester her er:

- hvis det er én type data (årstall, lag) i en bestemt rekkefølge → liste
- hvis vi kobler to ulike typer data (navn + nummer, svensk ord + japansk ord, student + godkjent) slik at vi kan slå opp på det ene og få det andre → ordbok

debugging-oppgave

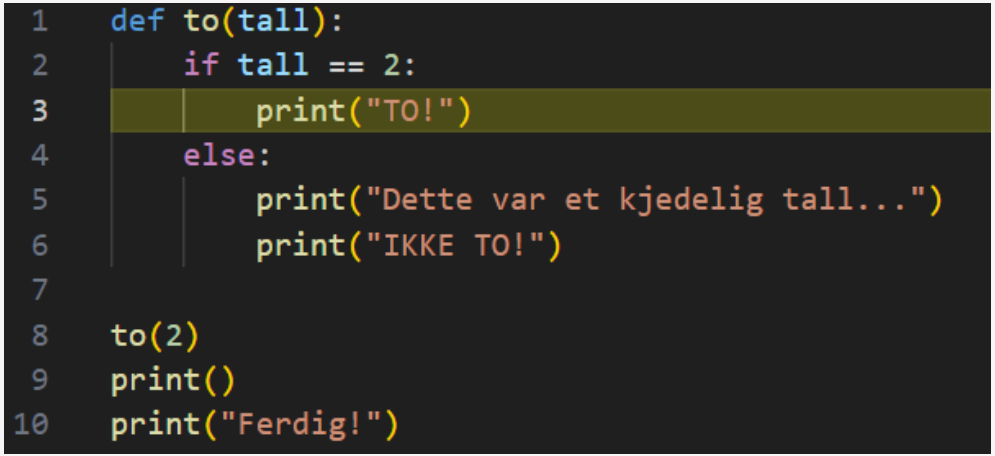
“du har laget en funksjon **doble** som dobler et tall, og tester den ved å la brukeren skrive inn et tall med **input** som skal dobles. her er tilstanden til variablene etter at programmet har kjørt. hva er feil her, og hvordan kan det fikses?”



løsningsforslag: problemet er at **input** returnerer **str**, og dermed vil operatoren ***** repetere strengen X ganger i stedet for å matematisk gange med X. løsningen er å bruke **int()** til å gjøre strengen om til et tall først. (selve funksjonen **doble** virker faktisk som den skal.)

kodeflyt-oppgave

“hvilken kodelinje er den neste som kjøres etter linjen som er markert i gult?”



```
1   def to(tall):
2       if tall == 2:
3           print("TO!")
4       else:
5           print("Dette var et kjedelig tall...")
6           print("IKKE TO!")
7
8   to(2)
9   print()
10  print("Ferdig!")
```

løsningsforslag: linje 9

prioritering av læremål

ikke fullstendig liste,
men forslag til et sted å starte

1) sentrale, fundamentale konsepter

- kunne ta vare på verdier med variabler
- bruke beslutninger: if, elif, else
- lese en feilmelding og informasjon fra debugger (python tutor)
- kodeflyt – hvilke linjer kjører i hvilken rekkefølge?
- evaluering av uttrykk
- objekter tilbyr tjenester (til programmet/andre objekter)
- løkker: for, while
- kunne bruke samlinger: lister og ordbøker
- funksjoner: bryte opp et program i deloppgaver
- hvordan bruke argumenter og parametre → funksjon
- hvordan bruke returverdier ← funksjon
- kunne skrive en klasse (med konstruktør og instansvariaber)
- kunne implementere et grensenitt (skrive klassens metoder) riktig
- kunne bruke egne og andres objekter (forstå og bruke grensesnitt)
- forstå og forklare forskjell på objekt og klasse

2) viktig å kunne og vite

- kunne lese inn fra terminalen
- kjenne til ulike datatyper og hva de gjør: int, float, str, bool
- hvordan referanser til objekter fungerer i maskinen
- hvordan løkker (spesielt for) fungerer sammen med samlinger
- nøstede samlinger: liste med lister, ordbok med lister, ...
- finne antall elementer i en samling
- lese fra og skrive til filer
- import fra klasser og pakker
- kunne forklare innkapsling og hvorfor det er nyttig

3) kjekt å ha vært borti / hørt om

- mengder
- range() – mer avansert bruk enn “repeter X ganger” i for-løkke
- split() for str
- count() for samlinger
- skrive tester med assert
- __str__ og andre magiske metoder


slidesgo