Egenerklæring

Jeg erklærer herved at besvarelsen som jeg leverer er mitt eget arbeid.

Jeg har ikke:

- samarbeidet med andre studenter
- brukt andres arbeid uten at dette er oppgitt
- brukt eget tidligere arbeid (innleveringer/ eksamenssvar) uten at dette er oppgitt

Om jeg har benyttet litteratur *ut over pensum*, vil en litteraturliste inneholde alle kilder jeg har brukt i besvarelsen og referanser vil vise til denne listen.

Jeg er kjent med at brudd på disse bestemmelsene er å betrakte som fusk og kan føre til annullert eksamen og/eller utestengelse.

Dersom du er usikker på om du kan stille deg bak erklæringen, se retningslinjer for bruk av kilder i skriftlige arbeider ved Universitetet i Bergen , og eventuelt ta kontakt med studieveileder/emneansvarli

Alle eksamensbesvarelser ved UiB blir sendt til manuell og elektronisk plagiatkontroll.

Merk: Ved å fortsette bekrefter jeg at jeg har lest erklæringen og at besvarelsen jeg leverer under denne eksamenen er mitt eget arbeid (og bare mitt eget arbeid), i full overensstemmelse med ovennevnte erklæringen.

For faglige spørsmål under eksamen

David Grellscheid er tilgjengelig på Discord for faglige spørsmål om eksamensinnhold.

I kanalen #exam_questions kan du klikke på konvolutt-ikonet. Det åpnes en privat "ticket"-kanal hvor du kan legge inn ditt spørsmål.

Kunngjøringer som er relevant for alle skjer i #announcements kanalen.

Alle andre kanaler blir ignorert.

Om du ikke er med i Discord ennå, kan du joine her: https://discord.gg/M7W92c37yw

For IKKE-faglige spørsmål under eksamen (praktiske/ tekniske spørsmål om eksamen eller Inspera) ta kontakt med oss i studieadministrasjonen ved Institutt for Informatikk. Det er 2 forskjellige kanaler, e-post og telefon:

Når du tar kontakt med oss i studieadministrasjonen (enten per e-post eller telefon) ber vi deg å ha:

1. kandidatnummeret ditt tilgjengelig
2. studentnummeret ditt tilgjengelig
3. kontaktinfoen din dersom vi må henvise saken din videre (telefonnummer/ e-post).

Per e-post: studieveileder@ii.uib.no

1. I emnefeltet skriv: INF100 – eksamen
2. I selve e-posten trenger vi ditt kandidatnummer og studentnummer
3. Beskriv så kort som mulig hva problemet er

Per telefon
55 58 41 59 – Eirik R. Thorsheim
55 58 92 26 - Linnin Gyberg

For generelle eksamensinformasjon har fakultetet laget en infoside: https://www.uib.no/matnat/56756/eksamen-ved-det-matematisk-naturvitenskapelige-fakultet#eksamen-og-korona-nbsp-ofte-stilte-sp-rsm-l

Eksamen består av en multiple-choice del som teller 25% og tre programmeringsoppgaver som teller 20%, 25% og 30%.

Generelle råd og kommentarer:

- Les nøye gjennom oppgavene før du begynner å svare.
- Dersom du ikke klarer å gi fullstendig svar til en oppgave, kan du likevel fortsette. Legg inn en kommentar som beskriver hva du skulle ha gjort i delen som mangler
- Koden din bør være leserlig og enkel å forstå. Velg gode variabelnavn og tydelig oppsett. Lag hjelpefunksjoner der de er nyttige. Opptil 5 poeng for hver av langsvarsoppgave kan trekkes fra i hver oppgave for uklar struktur.
- Syntes du at oppgaveteksten er uklar eller ufullstendig, må du lage dine egne forklaringer og gi disse i svaret som kommentar
- Bruk ikke for mye tid på multiple-choice delen! Det er ikke nok tid for å lime inn alle opsjoner i VSCode.
- Eventuelle filer vil også bli lastet opp til mittuib, så de kan enkelt hentes derfra

Lykke til!

1 Velg den passende datatype til de følgende uttrykk, der

- a = [1.1, 11]
- b = "11"
- c = 11
- d = b * a[1]

Finn de som passer sammen:

	list	bool	(-error-)	str	int	float
c == "11"	○	○	○	○	○	○
f"{a}"	○	○	○	○	○	○
b*c	○	○	○	○	○	○
a[1]	○	○	○	○	○	○
d	○	○	○	○	○	○
len(a)	○	○	○	○	○	○
a*b	○	○	○	○	○	○
[d]	○	○	○	○	○	○
a+b	○	○	○	○	○	○
c*a	○	○	○	○	○	○

Maks poeng: 5

2 Les inn hver linje fra en tekstfil og printe antallet bokstaver i linjen

filename = "foo.txt"
Velg alternativ (open, read, with, file) Velg alternativ (read(filename), filename, open(filename), with(filename)) Velg alternativ (to f:, with f:, as f:, from f:)

- for line in f:
  - Velg alternativ (line = line.split(), line = line.strip(), line = f.read(), line = f.readline()):
    - print(len(line))

Maks poeng: 2

3 Skriv inn navnet til en Exception-*type* slik at programmet printer "Unknown location". *Type* du velger må være så spesifikt som mulig:
f.eks. bruk "ZeroDivisionError", ikke bare "Exception", siden den kan også fange opp andre unntak.

```
location = {
  'Rick' : 'C-137',
  'Morty' : 'C500-A',
}
try:
```

- who = 'Summer'
- place = location[who]
- print(f"{who} is in {place}")

except [ ] :

- print("Unknown location")

except:

- print("Other error")

Maks poeng: 2

4 Velg slikt at alle sammenligninger er True. Dict xs ser sånn ut:

```
xs = {
'a' : 5,
'5' : 'hello',
7 : 3.1415,
5 : 7
}
```

Velg alternativ (xs[a], xs['a'], xs['5'], xs[5]) == 'hello'

'5' in Velg alternativ (xs.keys(), xs.items(), xs.values(), xs.setdefault())

Velg alternativ (xs[12], len(xs['5']), xs[5] + xs['a'], xs[7] + xs[5])== 12

Velg alternativ (len(xs[5]), xs[5], xs[7], len(xs['5'])) == xs['a']

Maks poeng: 2

5 Returner True hvis **kun** det første elementet fra *input* listen er oddetall, ellers returner False

def only_first_is_odd(input):

- x = input[0]
- if Velg alternativ (x // 2 == 1, x // 2 == 0, x % 2 == 0, x % 2 == 1) :
  - return Velg alternativ (False, True)
- for e in input[1:]:
  - if e % 2 == 1:
    - Velg alternativ (break, return False, return True, continue)
- return Velg alternativ (False, True)

Maks poeng: 2

6 Spør om 7 ord og lage en string fra deres første bokstavene

initials = ""

Velg alternativ (while True:, while False:, for initials in range(7):, for _ in range(7):)

- text = Velg alternativ (open("Text: "), read("Text: "), print("Text: "), input("Text: "))
- Velg alternativ (initials + text[0], initials = text[0], text += initials, initials += text[0])

print(f"The texts had Velg alternativ ({ text }, { initials }, (text), (initials)) as first letters.")

Maks poeng: 2

7 Velg de riktige verdiene til hver rad

a	b	a and b	a or b
True	True	Velg alternativ (True, False)	Velg alternativ (True, False)
True	False	Velg alternativ (True, False)	Velg alternativ (False, True)
False	False	Velg alternativ (True, False)	Velg alternativ (True, False)
5 > 3	5 > 7	Velg alternativ (False, True)	Velg alternativ (False, True)

Maks poeng: 2

8 Hvor ofte finnes det *letter* i strengen *word?*

def count(word, letter):

- ct = 0
- for i in word:
  - if i == letter:
    - Velg alternativ (ct = letter, ct += i, ct = 1, ct += 1)
- return Velg alternativ (ct, i, letter, word)

Maks poeng: 2

9 Velg de riktige linjene slik at resultatet av dette programmet er
A
B
C
og det kjører uten feil.

a = 450

Velg alternativ (if 'a' < 500:, if a < 500:, if a > 500:, if 'a' > 500:)

- print('A')

Velg alternativ (else:, elif a > 250:, elif a < 250:, if a > 250:)

- print('B')

Velg alternativ (if a % 10 != 0:, if a % 10 == 0:, elif a % 10 == 0:, elif a % 10 != 0:)

- print('C')

Velg alternativ (elif a < 500:, if a < 500:, if 'a' < 500:, elif:)

- print('D')

Maks poeng: 2

10 Skriv om denne løkken med *while* i stedet for *for*:

word = "blåbærsyltetøy"
sum = 0
for letter in word:

- if letter in "æøå":
  - sum += 1

---

word = "blåbærsyltetøy"
Velg alternativ (i = None, i = word, i = 0, i = len(word))
sum = 0
while Velg alternativ (i < len(word):, i < word:, sum < word:, sum < len(word):)

- Velg alternativ (sum = word[letter], letter = word[i], letter = word[sum], i = len(word))
- if letter in "æøå":
  - sum += 1
- Velg alternativ (i += 1, sum += 1, return sum, word += 1)

Maks poeng: 2

11 Velg slik at alle sammenligninger er True. Listen xs ser slik ut:
xs = ["hei", [12, 13], False, 1.3]

Velg alternativ (xs[-1], xs[1], xs[0], xs[0:1]) == 'hei'
'e' == Velg alternativ (xs[0:1], xs[0 1], xs[0][1], xs[0,1])
Velg alternativ (xs[0], xs[-2], xs[-3], xs[-1])== False
Velg alternativ (len(xs[1]), len(xs), len(xs[2]), len(xs[0])) == 2

Maks poeng: 2

12 Stortinget har 169 mandater som må fordeles mellom 19 valgdistrikter. Distrikter med høyere befolkningstall og/eller større areal får flere mandater enn de som er mindre.

Metoden er beskrevet slik i valgloven §11-3:

(2) Hvert valgdistrikts fordelingstall fastsettes ved at antallet innbyggere i valgdistriktet [...] adderes med antall kvadratkilometer i valgdistriktet multiplisert med 1,8.

(3) Hvert valgdistrikts fordelingstall divideres med 1 – 3 – 5 – 7 osv. De kvotienter som fremkommer, nummereres fortløpende. Representantplassene fordeles på valgdistriktene på grunnlag av de fremkomne kvotientene. Representantplass nr. 1 tilfaller det valgdistriktet som har den største kvotienten. Representantplass nr. 2 tilfaller det valgdistriktet som har den nest største kvotienten, osv. [...]

Denne tildelingsmetoden ligner nesten den for valgresultater som vi brukte i uke 8. Her er fasiten til denne oppgaven: https://folk.uib.no/dgr061/uke_08_oppg_5.py som du kan tilpasse hvis du vil, men da må ubrukte deler fjernes for å få en oversiktlig løsning.

CSV-filen https://folk.uib.no/dgr061/valgdistrikt_2020-01-01.csv inneholder data om distriktene: navn, antallet innbyggere, areal i kvadratkilometer. Filen er i formatet UTF-8.

Oppgave

- Lag en *funksjon* **mandatfordeling(filnavn, antall_mandater)** som tar som argument et filnavn til en CSV-fil med data om distriktene og antallet mandater som skal fordeles. Funksjonen skal returnere et *dict* (eller collections.*Counter)* som viser distriktnavn som *key* og antallet mandater som distriktet fikk, som *value.*

- I *hovedprogrammet* skal selve funksjonen kalles: **mandatfordeling("valgdistrikt_2020-01-01.csv", 169) .** Resultatet skal så printes fra høyest til færrest, med pen formatering, og med tallene justert til høyre:

```
Distrikt              Mandater
======================          (denne understreken har lengden 22)
Oslo                        20
Akershus                    19
...
...
...
```

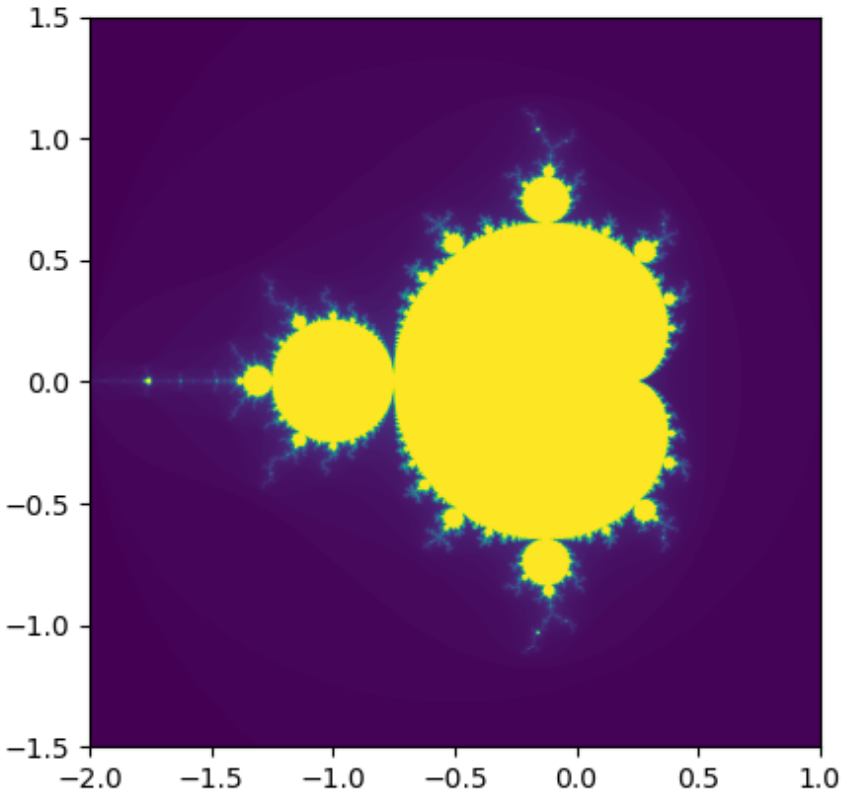
.

Skriv ditt svar her

Maks poeng: 20

13 Last ned filen https://folk.uib.no/dgr061/mandelbrot.py . Du skal **tilpasse filen** for å løse de etterfølgende oppgavene (det er 3 oppgaver).

Filen genererer dette Mandelbrot-fraktalet som et png-bilde. x-verdiene går fra -2 til 1, y-verdiene fra -1.5 til 1.5, og bildet har 1000x1000 piksler:



Oppgaver (lim inn **ett program** som gjennomfører alle deler**)**

(1) Istedenfor at alt skjer i hovedprogrammet, legg inn alt som er relevant i en funksjon **mandel(x, y, size, pixels, filename)** som tar inn 5 argumenter og skal bruke plt.savefig til å lagre en bildefil. Funksjonen skal ikke returnere noe.
Argumentene **x** og **y** er koordinater til det nedre venstre hjørnet,
size er avstanden fra x/ymin til x/ymax,
pixels er antallet piksler i hver retning, og
filename er navnet til filen som skal lagres.

(bildet ovenfor tilsvarer **mandel(-2., -1.5, 3., 1000, "mandelbrot.png")**)

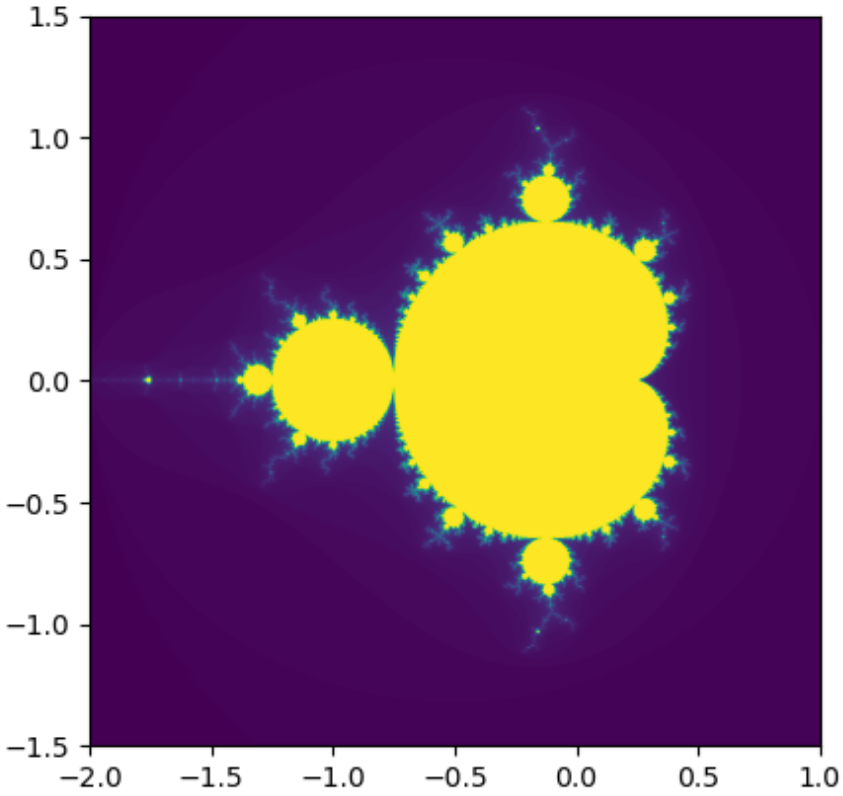
(2) Lag en funksjon
mandel_zoom(old_x, new_x, old_y, new_y, old_size, new_size, pixels, num_steps)
som ikke returnerer noe, men lagrer en sekvens av **num_steps** png-bilder med navn fra zoom_01.png frem til zoom_NN.png, der NN tilsvarer num_steps. Du skal kalle mandel()-funksjonen fra del 1 for hvert bilde:

Det første bildet skal tilsvare **mandel(old_x, old_y, old_size, pixels, "zoom_01.png")**

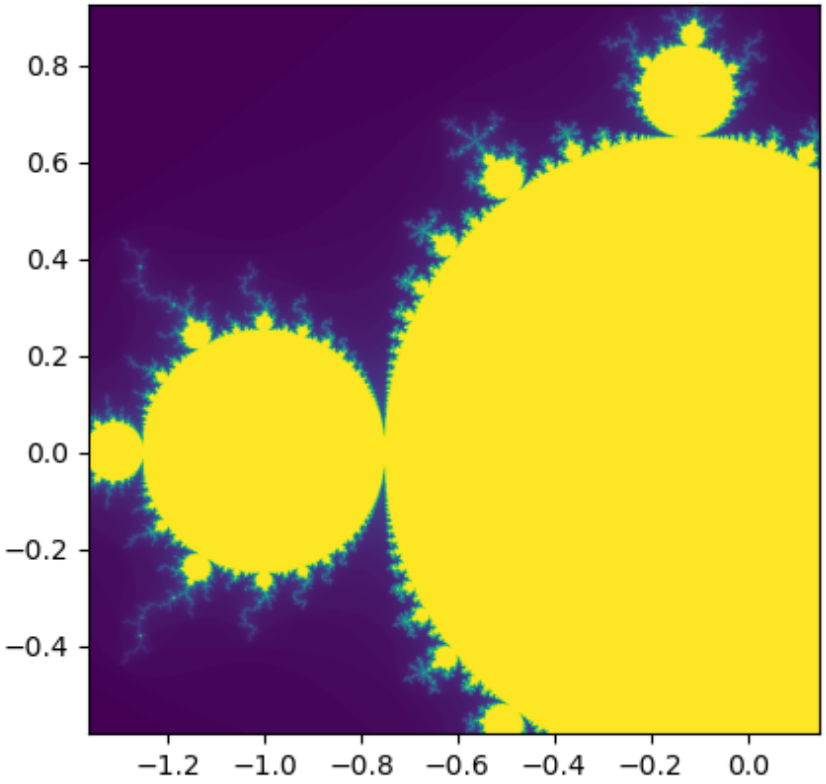
Det siste bildet skal tilsvare **mandel(new_x, new_y, new_size, pixels, "zoom_NN.png")**
Bildene imellom skal ta **x, y** og **size** fra en lineær interpolasjon mellom old_x og new_x, old_y og new_y, old_size og new_size. (Her kan du godt bruke np.linspace())

Eksempel: mandel_zoom(-2, -0.725, -1.5, 0.335, 3, 0.02, 1000, 3) lagrer 3 filer

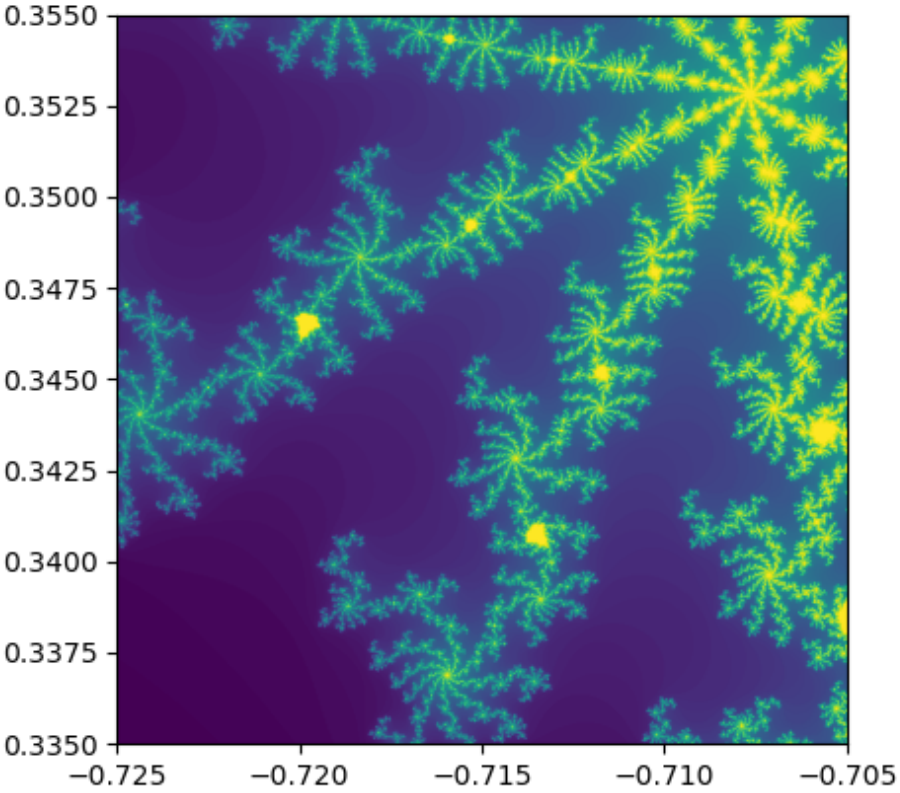
zoom_01.png



zoom_02.png


0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.2
−1.0
−0.8
−0.6
−0.4
−0.2
0.0

zoom_03.png


0.3550
0.3525
0.3500
0.3475
0.3450
0.3425
0.3400
0.3375
0.3350
−0.725
−0.720
−0.715
−0.710
−0.705

(3) Til slutt, skriv et nytt hovedprogram som spør brukeren med input() om antallet piksler og antallet zoom-bilder. Bruk exception-håndtering for å sjekke at brukeren skriver to heltall (integers). Om brukeren skriver noe annet enn tall, skal spørsmålet gjentas.

Så skal hovedprogrammet kalle funksjonen mandel_zoom(-2, -0.725, -1.5, 0.335, 3, 0.02, pixels, num_images) der **pixels** og **num_images** er de to tallene fra brukeren.

Lim inn den endelige koden som fullfører alle deler

Maks poeng: 25

14 Last ned filen https://folk.uib.no/dgr061/football.py . Du skal **tilpasse kun de markerte delene i filen** for å løse de etterfølgende deloppgavene. Denne oppgaven har 2 deler som kan løses hver for seg.

Vi skal se på noen resultater fra gruppefasen fra ulike fotball-VM. I hver gruppe finnes det 4 landslag som spiller 6 kamper mot hverandre. Kampresultatene er gitt som en liste av 4-tuples, med 2 strings og 2 int:

```
[
  ("Brazil", "Scotland", 2, 1),
  ("Morocco", "Norway", 2, 2),
  ("Scotland", "Norway", 1, 1),
  ("Brazil", "Morocco", 3, 0),
  ("Brazil", "Norway", 1, 2),
  ("Scotland", "Morocco", 0, 3),
]
```

For hvert lag kan vi nå lage en statistikk med 4 tall:

- **GF** (goals for) - antallet mål som laget har skåret
- **GA** (goals against) - antallet mål som ble skåret **mot** laget
- **GD** (goal difference) - differansen mellom GF og GA, alltid lik GF-GA
- **PT** (points) - poengsummen: 3 poeng til kampvinneren, 1 poeng til hvert lag om kampen er uavgjort

Denne statistikken skal lagres i en datastruktur som er et dict av dicts. Til eksemplet ovenfor ser det sånn ut:

```
{
  'Brazil': {'GF': 6, 'GA': 3, 'GD': 3, 'PT': 6},
  'Scotland': {'GF': 2, 'GA': 6, 'GD': -4, 'PT': 1},
  'Morocco': {'GF': 5, 'GA': 5, 'GD': 0, 'PT': 4},
  'Norway': {'GF': 5, 'GA': 4, 'GD': 1, 'PT': 5},
}
```

Denne strukturen er **ikke sortert** ennå, en tilfeldig rekkefølge er greit!

Deloppgave 1 (make_stats)

Lag en funksjon **make_stats(matches) ,** der **matches** er en liste med kampresultater i det formatet vist ovenfor. Funksjonen skal returnere et dict med statistikken i det nevnte dict-formatet. Det finnes et skjelett til make_stats() allerede i filen, med en liten if-test som sjekker funksjonen din.

Selv om du ikke klarer deloppgave 1, kan du fjerne 'exit()' på linje 44, og prøve å løse deloppgave 2.

Deloppgave 2 (compare)

Her skal vi sortere lagene i en tabell for gruppen. Print-funksjoner osv. er allerede klare, det eneste som mangler er en **compare-**funksjon som avgjør rekkefølgen til lagene. Avgjørelsen skjer med flere regler, der vi bruker den neste når alle tidligere regler ikke gir én vinner:

(1) Større poengsum PT
(2) Større måldifferanse GD

(3) Flere skårete mål GF
(4) Vinneren av den direkte kampen mellom de to lagene
(vi ignorer en situasjon hvor det er flere enn to lag som er helt like når vi kommer til denne regelen)
(5) Trekking av lodd

Lag en funksjon **compare(matches, data, a, b)** der
matches er listen til kampresultatene i formatet ovenfor,
data er et ferdig dict med statistikk i formatet ovenfor,
a og **b** er to strenger med **navnene** til 2 landslag

(vi tar med *data* som argument slik at vi kan ignorere funksjonen i del 1.
Vi kan alltid anta at *data* == make_stats(*matches*)

Funksjonen skal returnere tallet **1** om lag **a** ligger høyere i tabellen enn lag **b**
Funksjonen skal returnere tallet **-1** om lag **b** ligger høyere i tabellen enn lag **a**
Funksjonen skal printe **f"LOTTERY {a} {b}"** og returnere tallet **0** om situasjonen er uavgjort mellom **a** og **b**

Det finnes et skjelett til compare() allerede i filen, med en rekke tester som sjekker funksjonen din.

Eksempel output

Om alt er bra i begge deler, ser outputtet slik ut (husk å fjerne exit() i linje 44). *Formateringen i kolonner kommer ikke frem pent i Inspera:*
---

make_stats looks good, you can remove the exit() line now!

```
            GF  GA  GD   PT
Brazil       6   3   3    6
Norway       5   4   1    5
Morocco      5   5   0    4
Scotland     2   6  -4    1
--------------------
This looks good!


            GF  GA  GD   PT
Nigeria      6   2   4    6
Bulgaria     6   3   3    6
Argentina    6   3   3    6
Greece       0  10 -10    0
--------------------
This looks good!


            GF  GA  GD   PT
Mexico       3   3   0    4
```

```
Ireland          2   2   0     4
Italy            2   2   0     4
Norway           1   1   0     4
--------------------
This looks good!


LOTTERY Ireland Netherlands
                GF  GA  GD    PT
England          2   1   1     5
Ireland          2   2   0     3
Netherlands      2   2   0     3
Egypt            1   2  -1     2
--------------------
This looks good!
```

Legg inn kode her, ikke output

Maks poeng: 30