NTNU

Kunnskap for en bedre verden

KANDIDAT

Østby Sindre (sindos)

PRØVE

TDT4110 Insperaøving 1 - B

Emnekode	--
Vurderingsform	--
Starttid	28.09.2020 06:00
Sluttid	02.10.2020 15:00
Sensurfrist	--
PDF opprettet	01.10.2020 03:02

Seksjon 1

Oppgave	Tittel	Oppgavetype
i	Forside	Dokument

Teori

Oppgave	Tittel	Oppgavetype
i	Teorispørsmål	Dokument
1	Lyd	Langsvar
2	Tekstkoding	Langsvar

Variabeltyper

Oppgave	Tittel	Oppgavetype
3	Variabeltyper	Paring

Pseudokode

Oppgave	Tittel	Oppgavetype
i	Pseudokode	Dokument
4	Pseudokode - Del 1	Tekstfelt
5	Pseudokode - del 2	Tekstfelt
6	Pseudokode - del 3	Tekstfelt

Diverse programmering

Oppgave	Tittel	Oppgavetype
7	Minste tall	Programmering
8	Fotballfan	Programmering
9	Gangetabell	Programmering

Logiske operatorer

Oppgave	Tittel	Oppgavetype

10	Logiske operatorer - del 1	Programmering
11	Logiske operatorer - del 2	Programmering

Telle bokstaver

Oppgave	Tittel	Oppgavetype
12	Telle bokstaver - del 1	Programmering
13	Telle bokstaver - del 2	Programmering

Måskråke

Oppgave	Tittel	Oppgavetype
14	Maskråke - del 1	Programmering
15	Maskråke - del 2	Programmering
16	Maskråke - del 3	Programmering

Trekant

Oppgave	Tittel	Oppgavetype
17	Trekant - del 1	Programmering
18	Trekant - del 2	Programmering
19	Trekant - del 3	Programmering

Seksjon 10

Oppgave	Tittel	Oppgavetype
20	Mortens kube	Programmering

1

Lyd

Forklar kort hvordan analog lyd kan overføres til og representeres i en datamaskin:

Skriv ditt svar her

En mikrofon fanger opp lydbølgene og gjør de om til resistans i en ledning, dette blir en bølgeform, og er notert kontinuerlig. Lydnivåene testes en viss antall ganger i seundet og blir skrevet ned digitalt til en blokknotasjon. Lyden noteres som reistans pr. sample og kan tenkes på som å gjøre bølgeformen om til en rekke rektangler hvis det hadde blit nortert som en graf. I en datamaskin vil det representereres i et filformat som har en bestemt måte å skive opp hvordan du konverterer frem og tilbake til resestansen og som oftes også inneholder kompresjon (ikke nødvendig)

2

Tekstkoding

Forklar kort hvorfor tekstkodingen UTF-8 er bedre egnet for koding av tekst på websider enn det ASCII er:

Skriv ditt svar her

UTF-8 har fordelen av at alle blokker noteres i 8 bit, der ASCII noteres i 7 eller fler. UTF-8 baserer seg på at de 7 første bit inneholder informasjon og den 8. sier om du skal fortsette å lese eller om tegnet er ferdig. Dette gjør at du kan ha svært mange flere tegn enn du kan i ASCII. Det er også mer egnet fordi det er blit et standard-språk og ca. alle moderne enheter kan kjøre koden.

3

Variabeltyper

Anta at du ønsker å samle inn data om vennene dine. Dataen du ønsker å lagre er navnet deres, fødselsåret deres, hvor mange dager de bruker på å spise en 200g plate melkesjokolade (inkludert halve dager), og om de liker sjokoladekake eller ikke.

Eksempel på data er:

Anna of Arendelle er født i 2013 og bruker halvannen dag på å fullføre en sjokoladeplate. Hun er en stor fan av sjokoladekake.

Hvilken variabeltype (integer/float/string/boolean) ville du brukt for de ulike dataene for en gitt person?

Velg de som passer sammen:

	Integer	Float	String	Boolean
Navn	○	○	●	○
Fødselsår	●	○	○	○
Dager brukt på å fullføre en sjokoladeplate	○	●	○	○
Liker / Liker ikke sjokoladekake	○	○	○	●

Erstatt denne teksten med ditt innhold...

4 Pseudokode - Del 1

Forklar hva denne funksjonen gjør:

```
def unknown_func(n):
    if n is greater than 1:
        for i=2 to n/2+1:
            if n modulo i equals 0:
                return False
    return True
```

Skriv inn svar her

hvis det retunerer true, er det primtall. his det returnerer false er det ikke det

5 Pseudokode - del 2

Forklar kort hva denne funksjonen gjør:

```
x = 3
while x:
    write x number of "#" to screen
    subtract 2 from x
```

Skriv ditt svar her

koden vil først skrive 3 x '#', så 1 x '#' og så bli derdig fordi x < 0

6 Pseudokode - del 3

Forklar hva som skjer når *unknown_func2(1234)* kalles:

```
def unknown_func2(a):
    r = 0
    while a is greater than 0:
        s = a modulo 10
        r = r plus s
        a = (a minus s) divided by 10
    return r
```

Skriv ditt svar her

funksjonen vil føre det siste sifferet i tallet (i titalsystemet) til s, legge det til r, fjærne det siste sifferet fra a og dele på ti (gjøre 1234 til 123) dette repeteres til a er 0 og vil gi ut summen av sifferene til tallet i titallssystemet. r = sum av siffer i titallssystemet

7 Minste tall

Lag et program som spør bruker om å oppgi et tall (både flyttall og heltall er akseptert) 5 ganger. Deretter skal programmet printe hvilke av de 5 tallene som var minst.

Eksempel på kjøring:

```
Skriv et tall: 10
Skriv et tall: 23.5
Skriv et tall: 0.991
Skriv et tall: -13
Skriv et tall: 42
Det minste tallet du skrev inn var -13.0.
```

Skriv ditt svar her

```
tallarray = []
for i in range(5):
    tall = int(input('Skriv et tall: '))
    tallarray.append(tall)

print(min(tallarray))
```

8 Fotballfan

Se for deg at du er det eneste fotballfanet i vennegjengen og at det er fotballkamp til helgen hvor du heier på hjemmelaget. Siden du ikke ønsker å plage vennene dine med enda mer fotballsnakk, bestemmer du deg for å lage et lite program som du kan interagere litt med etter kampen. Dette programmet skal ta inn en streng fra bruker som beskriver resultatet av kampen (h=hjemmeseier, b=borteseier, og u=uavgjort), og gi tilbakemelding på resultatet. Programmet skal akseptere både store og små bokstaver og fremdeles gi korrekt tilbakemelding. Videre skal det gi en unik tilbakemelding om bruker ikke oppgir en av de tre bokstavene h, b eller u.

Lag dette programmet.

Eksempler på kjøring:

```
Hva ble resultatet? h
Yay, det ble hjemmeseier!
Hva ble resultatet? b
Oufh, så trist å tape på egen bane.
Hva ble resultatet? B
Oufh, så trist å tape på egen bane.
Hva ble resultatet? u
Snufs, men kunne vært verre.
Hva ble resultatet? grynt
Gikk det såpass dårlig ja? Ønsker ikke å snakke om det engang...
```

Skriv ditt svar her

```
resultat = input('Hva ble resultatet? ')
resultatet = resultat.lower()
print(resultatet)
if resultatet == 'h':
    print('Yay, det ble hjemmeseier!')
elif resultatet == 'b':
    print('Oufh, så trist å tape på egen bane.')
elif resultatet == 'u':
    print('Snufs, men kunne vært verre.')
else:
    print('Gikk det såpass dårlig ja? Ønsker ikke å snakke om det engang...')
```

9 Gangetabell

I denne oppgaven skal du lage en funksjon, *print_table(n)*. Denne funksjonen tar inn parameteren *n*, og skriver ut en *nxn* gangetabell, som i eksempelet under.

Hint: *Nøstede løkker*

Eksempel på kjøring av *print_table(4)*:

```
>>> print_table(4)

 1 2 3 4
 2 4 6 8
 3 6 9 12
 4 8 12 16
```

Skriv ditt svar her

```
def print_table(n):
    for tall_1 in range(1, n + 1):
        for tall_2 in range(1, n + 1):
            print(tall_1 * tall_2, end=' ')
        print()
```

10 Logiske operatorer - del 1

Lag et program der du spør brukeren om en måned (string) og et årstall (integer). Basert på denne informasjonen skal programmet skrive ut hvor mange dager det er i måneden, ved bruk av logiske operatorer.

NB! Du må ta hensyn til skuddår, da februar har 29 dager. Du kan bruke den følgende forenklede definisjonen på et skuddår: Et skuddår er alle år der årstallet er delelig på 4.

Du kan anta at både måned er gitt som en korrekt streng, og år som et positivt heltall.

Januar	31
Februar	28 (29)
Mars	31
April	30
Mai	31
Juni	30
July	31
August	31
September	30
Oktober	31
November	30
Desember	31

Eksempel:

```
>>> %Run 'Logiske operatorer

   Skriv inn en måned: mars
   Skriv inn et år: 2020
   31
```

Skriv ditt svar her

```
1  month = input('Skriv inn en måned: ')
2  year = int(input('Skriv inn et år: '))
3
4  if month == 'januar' or month == 'mars' or month == 'mai' or month == 'juli' or month == 'august' or
       month == 'oktober' or month == 'desember':
5      print(31)
6  elif month == 'april' or month == 'juni' or month == 'september' or month == 'november':
7      print(30)
8  elif year % 4 == 0:
9      print(29)
10 else:
11     print(28)
```

11 Logiske operatorer - del 2

Utvid programmet fra den forrige oppgaven. Det nye programmer setter større krav til **inputen**. Om brukeren skriver en **ugyldig måned eller et ugyldig år** skal programmet **spørre brukeren på nytt**. Gyldige år er alle år fra og med år 0 til og med år 2020. Programmet skal tillate at man skriver inn måneden med både store og små bokstaver.

Du trenger ikke håndtere tilfeller hvor år ikke gis som et heltall.

Eksempel:

```
>>> %Run 'Logiske operatorer b.py'

   Skriv inn en måned: MaRs
   Skriv inn et år: 2030
   Ugyldig input! Prøv på nytt!
   Skriv inn en måned: MaRs
   Skriv inn et år: 2020
   Det er  31  dager i denne måneden.
```

Skriv ditt svar her

```
1  while true:
2      month = input('Skriv inn en måned: ').lower()
3      year = int(input('Skriv inn et år: '))
4      if year > 2020 or year < 0
5          print('Ugyldig input! Prøv på nytt!')
6      elif month == 'januar' or month == 'mars' or month == 'mai' or month == 'juli' or month ==
           'august' or month == 'oktober' or month == 'desember':
7          print(31)
8          exit(0)
9      elif month == 'april' or month == 'juni' or month == 'september' or month == 'november':
10         print(30)
11         exit(0)
12     elif year % 4 == 0:
13         print(29)
14         exit(0)
15     elif: month == 'februar'
16         print(28)
17         exit(0)
18     else:
19         print('Ugyldig input! Prøv på nytt!')
```

12

Telle bokstaver - del 1

Lag en enkel funksjon *correct_word(word)* som tar inn en streng *word* og returnerer ***True*** dersom strengen kun inneholder bokstaver og ***False*** hvis strengen inneholder tegn som ikke er bokstaver, slik som #!.,_: (mellomrom/whitespace).

Tips: Funksjonen *isalpha()* kan komme til nytte.

Eksempel:

```
print(correct_word("Ent3r"))
print(correct_word("TDT4109"))
print(correct_word("Kybernetikk"))
```

```
>>> %Run telle-bokstaver-1.py

  False
  False
  True
```

Skriv ditt svar her

```
1 def correct_word(word):
2     if isalpha(word) == 1:
3         print(True)
4     else:
5         print(False)
```

13

Telle bokstaver - del 2

Vi ønsker å vite hvor mange bokstaver et ord består av.

Lag en funksjon *count_letters(word)* som tar inn en streng *word* og **returnerer antall bokstaver i ordet.**

NB! Sjekk at ordet bare består av bokstaver. Om ordet inneholder andre elementer, skal funksjonen returnere -1.

Du kan bruke *correct_word(word)* fra forrige oppgave, uansett om du fullførte den eller ikke.

Eksempel:

```
print(count_letters("Ouagadougou"))
print(count_letters("ITGK<3"))
```

```
>>> %Run telle-bokstaver-2.py

  11
  -1
```

Skriv ditt svar her

```
1 def count_letters(word):
2     if isalpha(word) == 1:
3         print(len(word))
4     else:
5         print(-1)
```

14 Maskråke - del 1

Lag funksjonen *ask_question* som tar inn et spørsmål i form av en streng, stiller dette spørsmålet til bruker, og returnerer svaret fra bruker.

Eksempel på kjøring:

```
answer = ask_question("Hvor langt er det igjen? ")
print(answer)

# Running the codesnippet above,
# will result in the following output if the user's answer is "3 timer":

Hvor langt er det igjen? 3 timer
3 timer
```

Skriv ditt svar her

```
def ask_question(question):
    return input(question)
```

15 Maskråke - del 2

Lag funksjonen *spam_with_questions* som tar inn et spørsmål i form av en streng. Funksjonen skal stille dette spørsmålet om og om igjen, helt til bruker svarer med "stopp". Ordet "stopp" skal ikke være case-sensitivt, dvs. at alle kombinasjoner av små og store bokstaver skal fungere. F.eks. "STOPP", "StOpp" og "stopp".

Du kan benytte deg av funksjonen *ask_question* fra forrige oppgave, *Maskråke - del 1*, om du ser behov, uansett om du klarte oppgaven eller ikke.

Eksempel på kjøring:

```
>> spam_with_questions("Hvor langt er det igjen? ")
Hvor langt er det igjen? 3 timer
Hvor langt er det igjen? Fortsatt 3 timer
Hvor langt er det igjen? 3 timer
Hvor langt er det igjen? Det går ikke så mye raskere av at du spør hvert andre sekund
Hvor langt er det igjen? Du kan jo gjette?
Hvor langt er det igjen? Vær så snill, stopp
Hvor langt er det igjen? Stopp
```

Skriv ditt svar her

```
def ask_question(question):
    return input(question)

def spam_with_question(question):
    a = 0
    while a != 'stopp':
        a = input(str(ask_question(question))).lower()
```

16 Maskråke - del 3

Lag funksjonen *energy_efficient_spamming* som tar inn to strenger. Den første strengen er et spørsmål til bruker som stilles én gang, mens den andre strengen er et oppfølgingsspørsmål som stilles om og om igjen helt til bruker svarer "stopp", "STOPP", "sToPP" eller en annen variasjon av små og store bokstaver for ordet "stopp".

For at noen av spørsmålene i det hele tatt skal bli spurt, må det første spørsmål være lengre enn det andre. Dette fordi noe annet ikke ville vært energibesparende.

Du kan benytte deg av funksjonene *ask_question* og *spam_with_questions* fra oppgavene *Maskråke - del 1* og *del 2*, om du ser behov, uansett om du klarte oppgaven eller ikke.

Eksempel 1 på kjøring:

```
>> energy_efficient_spamming("Hvor langt er det igjen? ", "Hva med nå da? ")
Hvor langt er det igjen? 3 timer
Hva med nå da? Fortsatt 3 timer
Hva med nå da? ...
Hva med nå da? stopp
```

Eksempel 2 på kjøring:

```
>> energy_efficient_spamming("Hvor langt er det igjen? ", "Hva langt er det igjen nå da? ")


# Her blir ikke bruker spurt noe (ingenting skrives til skjerm), siden den andre strengen er lengre enn den første
```

Skriv ditt svar her

```
def ask_question(question):
    return input(question)

def spam_with_question(question_1, quwstion_2):
    a = 0
    if len(question_1) > len(question_2)
        input(ask_question(question_1))
        while a != 'stopp':
            a = input(str(ask_question(question_2))).lower()
```

17 Trekant - del 1

Lag en funksjon *triangle(h)* som tar inn en høyde *h* (integer) og skriver ut en rettvinklet trekant som vist i eksempelet under.

Eksempel med *h=5*:

```
>>> triangle(5)
*
* *
* * *
* * * *
* * * * *
```

Skriv ditt svar her

```
def triangle(n):
    for i in range(1, n + 1):
        print(i)
```

18

Trekant - del 2

Lag en funksjon *flipped_triangle(h)* som tar inn en høyde *h* (integer) og printer ut en rettvinklet trekant som er opp ned i forhold til trekanten funksjonen *triangle(h)* printet i forrige oppgave.

Eksempel med *h=5*:

```
>>> flipped_triangle(5)
* * * * *
* * * *
* * *
* *
*
```

Skriv ditt svar her

```
1
```

19

Trekant - del 3

Lag en funksjon *isosceles_triangle(h)* som tar inn en høyde *h* (integer) og skriver ut en likebeint trekant som hvist i eksempelet under.

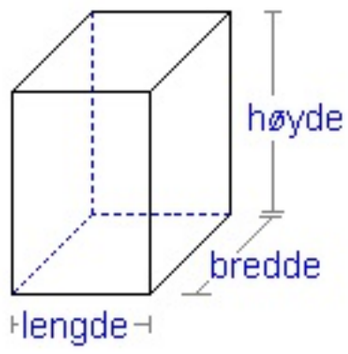
Eksempel med *h=5*:

```
>>> isosceles_triangle(5)
    *
   * *
  * * *
 * * * *
* * * * *
```

Skriv ditt svar her

```
1
```

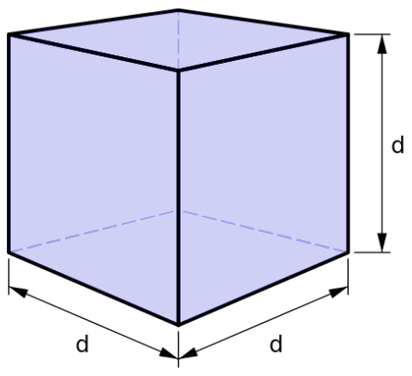
Rett firkantet prisme



Figuren over viser ett rett, firkantet prisme

Kube

En kube er et rett, firkantet prisme som har *lengde=bredde=høyde*.



Figuren over viser en kube med $l = b = h = d$, altså har lengden, bredden og høyden verdi d.

Volumet av en kube er definert som $V = d \cdot d \cdot d = d^3$.

For å finne det største volumet en kube som er plassert inne i et rett, firkantet prisme kan ha, må du finne den minste verdien av lengden, bredden og høyden til prismet, og sette d til denne verdien.
Har for eksempel prismet dimensjoner $l = 6, b = 3, h = 4$ får vi at $d = 3$. Dette gir et volum på $V = d^3 = 3^3 = 27$.

20 Mortens kube

Morten liker å grave hull, med visse begrensninger. Han graver bare hull som har form som et *rett, firkantet prisme*, og han vil ikke grave hull der lengde, bredde eller høyde er like. Altså må lengden, bredden og høyden ha forskjellige verdier.

Morten vil grave et nytt hull, men trenger din hjelp. Han vil 3D-printe en *kube* som skal få plass i hullet. Kuben skal ha et så stort volum som mulig, men skal passe i hullet uten å stikke opp av bakken.

Oppgaven din er å lage et program som hjelper Morten å overholde kravet sitt, i tillegg til å regne ut det største mulige volumet til en kube som passer i hullet. Det du må gjøre er å:

- Spørre brukeren om verdier for lengde, bredde, og høyde, og lagre disse i variabler.
- Sjekke om disse overholder Mortens krav, altså at ingen av dem er like.
    - Om kravet **ikke** er overholdt, skal du skrive ut en beskjed om at hullet ikke kan graves, før du ber brukeren om input igjen. Dette skal fortsette helt til kravet er overholdt.
    - Om kravet er overholdt, skal du regne ut volumet til den største kuben som får plass i hullet uten å stikke opp av bakken. Dette volumet skal skrives til skjerm.

Du trenger ikke å håndtere feil som at inputen ikke er et tall. I tillegg trenger du ikke å regne ut en enhet på volumet.

Tips: Det kan være lurt å bruke funksjoner om du ser at noe kode må brukes flere steder.

Eksempel:

```
Lengde: 4.2
Bredde: 8
Høyde: 8
Morten er ikke fornøyd, prøv igjen
Lengde: 6.7
Bredde: 4.2
Høyde: 7
Den største kuben vil ha et volum på 74.09.
```

Skriv ditt svar her

```python
def volum(bredde, lengde, hoide):
    a = []
    a.append(lendge, bredde, hoide)
    d = min(a)
    return d**3

while true:
    lengde = float(input('Lengde: '))
    bredde = float(input('Bredde: '))
    hoyde = float(input('Høyde: '))
    if hoyde != lengde and hoyde != bredde and bredde != lengde:
        print(f'Den største kuben vil ha et volum på {volum(bredde, lengde, hoyde)}')
        exit(0)
    else:
        print('Morten er ikke fornøyd, prøv igjen')
```