LF Oving 1
Harald Minde Hansen authored 1 month ago

75551e36

James Bond lf.ipynb 6.84 KB

a)

Denne løsningen klarer seg med det som er pensum som er undervist før øvingen, men har en liten skjønnhetsfeil (det kommer .0 bak tall som skal være heltall)

```
tall = float(input("Tallet som skal avrundes: "))
d = int(input("Ønsket antall desimaler: "))
faktor = 10 ** d
resultat = int(tall * faktor + 0.5) / faktor
print("Avrundet til", d, "desimaler:", resultat)
```

For å få vekk skjønnhetsfeilen er det mest hensiktsmessig å bruke en if-setning:

```
tall = float(input("Tallet som skal avrundes: "))
d = int(input("Ønsket antall desimaler: "))
resultat = int(tall + 0.5)
faktor = 10 ** d
resultat = int(tall * faktor + 0.5) / faktor
if d <= 0:  # skal ende med et heltall
    resultat = int(resultat)
print("Avrundet til", d, "desimaler:", resultat)
```

b) Ideen her er å sette sammen heltallsdel og desimaldel til ett heltall og gjøre avrundingen på det (siden heltall alltid lar seg representere eksakt), da kan vi bruke round() med negativt antall desimaler for å kvitte oss med passende mange desimaler. Hvor mange kan regnes ut som differansen mellom antall desimaler vi har (lengden av desimalstrengen) og antallet vi ønsker. De overflødige desimalene kuttes vekk med heltallsdivisjon og til slutt må punktumet settes tilbake på rett plass (ved hjelp av heltallsdivisjon og modulo).

```
hel = input("Oppgi heltallsdelen av tallet (det foran punktum): ")
des = input("Oppgi desimaldelen av tallet (det bak punktum): ")
inn_des = len(des) #lengden av strengen, dvs antall desimaler nå
tall = int(hel + des)
ant = int(input("Oppgi ønsket antall desimaler i avrunding: "))
kuttes = max(inn_des - ant, 0) #hvor mange desimaler skal kuttes
avr = round(tall, -kuttes)
# bruker heltallsdivisjon for å finne sifrene som er heltallet
nytt_heltall = avr // 10 ** inn_des
hel_ut = str(nytt_heltall)
# bruker modulo for å finne sifrene som er desimaler
# og deretter heltallsdivisjon for å kutte desimaler som skal bort
nytt_destall = (avr % 10**inn_des) // 10**kuttes
# setter sammen tallet, "." tilbake på riktig plass
# bool(ant) blir False hvis ant er 0, da tom streng for desimaldel
tall_ut = str(nytt_heltall) + ("." + str(nytt_destall)) * bool(ant)
d = "desimal" + "er" * bool(ant-1)
print(hel + "." + des, "avrundet til", ant, d, "blir", tall_ut)
```

c) Denne kan løses på mange måter. En mulighet er å bruke for-løkke og if-setning som vist under.

```
navn= input("Jeg heter: ")
sist = len(navn)
start = sist
for i in range(len(navn)):
    if navn[i]==" ":
        start = i + 1

print("The name is", navn[start:sist]+", " + navn)
```

En annen mulighet er å bruke strengmetoden split():

```
navn = input("Jeg heter: ")
liste = navn.split()
print("The name is", liste[-1] + ",", navn)
```

Forklaring av denne løsningen: Linje 2, split() gir en oppdeling av tekststrengen der det er mellomrom, med navn "Grace Murray Hopper" vil vi få liste med ordene "Grace", "Murray", "Hopper".

Linje 3: liste[-1] vil være bakerste element i variabelen liste, altså "Hopper" med eksemplet gitt ovenfor.

Begge de ovenstående løsningene vil imidlertid få problemer med navn med påheng som Jr, Sr, d.y., III osv., såvel som med navn med preposisjoner som Von, De, ... Hvis man skal ta høyde for dette, blir løsningen noe mer innfløkt, nedenfor en mulighet som en videreutvikling av løsningen med split() ovenfor.

Det fins selvsagt flere mulige elementer man kunne ha lagt til både i PREP og EXT. Løsningen tar ikke høyde for at noen kan ha flere tillegg bak navnet (f.eks. både Jr og III i samme navn), eller navn hvor det fins flere enn ett mellomord som skal være del av etternavnet (f.eks. Alan van der Aalst).

Dette kunne f.eks. ha vært taklet ved å kjøre både sjekken vs. EXT og sjekken vs. PREP i while-løkker heller enn som engangstester, koden ville da ha blitt noe mer komplisert enn den er nå.

```
PREP = {"De", "de", "Di", "di", "von", "Von", "van", "Van", "St", "Saint"}
EXT = {"Jr", "jr", "Sr", "sr", "d.e.", "d.y.",
       "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"}

navn = input("Jeg heter: ")
liste = navn.split() # får ei liste med alle delnavn i navnet

if liste[-1] in EXT: #bakerste ord er ikke et navn
    i_enavn = -2 # etternavnet da nest bakerst i lista
else:
    i_enavn = -1 # etternavnet er bakerst i lista
etternavn = liste[i_enavn]

if len(liste) > abs(i_enavn - 1): # det fins mellomnavn
    mellom = liste[i_enavn - 1] # mellomnavnet er like foran etternavn
    if mellom in PREP: # mellomnavn skal inngå i etternavnet
        etternavn = mellom + " " + etternavn # putter mellomnavn foran

print("The name is", etternavn + ",", navn)
```