Ohjelmoinnin peruskurssi Y1

CS-A1111

4.10.2023

Oppimistavoitteet: tämän luennon jälkeen

- Tiedät, miten ohjelman toimintaa voi tutkia ja ohjelmassa olevia virheitä etsiä debuggerin avulla.
- Osaat käsitellä useita yhteenkuuluvia arvoja yhden muuttujan kautta käyttämällä hyväksi listoja.
- Voit luennon aikana lähettää kysymyksiä ja kommentteja sivulla `http://presemo.aalto.fi/y1syksy2023`

Debuggeri

- Tyypillinen tilanne: ohjelma on kirjoitettu, Python-tulkki ei valita virheistä, mutta ohjelma toimii jotenkin väärin.
- Ohjelman toimintaa voi tutkia ja virhettä yrittää etsiä *debuggerin* avulla.
- Debuggerin avulla ohjelmaa voi ajaa käsky kerrallaan ja tutkia samalla ohjelman käyttämien muuttujien arvoja.
- Ohjelmaan voi asettaa pysähdyspisteitä (breakpoint), joissa ohjelma pysähtyy.
- PyCharmiin kuuluu debuggeri.

Debuggerin käytttö PyCharmissa

- Ennen debuggerin käynnistymistä kannattaa asettaa ensimmäinen pysähdyspiste. Valitaan haluttu rivi koodista, painetaan rivin vasemmassa reunassa hiiren painiketta – reunaan ilmestyy punainen pallo.
- Tämän jälkeen debuggerin voi käynnistää Run-valikosta tai valitsemalla hiiren oikealla painikkeella avautuvasta valikosta `Debug 'moduulin_nimi'`.
- PyCharm ajaa ohjelmaa ensimmäiseen pysähdyspisteeseen saakka ja siirtyy Debugger-näkymään (ellei se ole jo käytössä).
- Tämän jälkeen ohjelmassa voi edetä rivi kerrallaan Debugger-välilehden yläreunan painikkeilla.

Debuggerin käytttö PyCharmissa, jatkoa

- Jos rivillä on funktion kutsu, ja haluaa siirtyä funktion sisälle, on valittava `Step Into` tai `Step Into My Code`. Jos ei halua siirtyä funktion sisälle, voi valita `Step Over`.
- Jos haluaa siirtyä pois funktiosta takaisin sitä kutsuneeseen kohtaan, voi valita `Step Out`.
- Jos ohjelma kysyy syötettä käyttäjältä, voi olla tarpeen vuorotella ikkunan alaosan Console- ja Debugger-välilehtien välillä.
- Muuttujien arvot näkyvät joka vaiheessa ikkunan alaosassa ja editorissa ohjelmakoodin vieressä.
- Ideana on ajaa ohjelmaa käsky kerrallaan ja katsoa, miten ohjelman suoritus etenee ja muuttujien arvot muuttuvat.
- Kun jossain vaiheessa arvo ei vastaa odotettua tai siirrytään eri riville kuin pitäisi, voi miettiä tarkemmin, miksi näin tapahtuu. Usein tämä johtaa virheen jäljille.
- Ohjelman ajamisen debuggaustilassa voi lopettaa painamalla vasemmassa reunassa olevaa punaista neliötä.

Listat

- Esimerkki: halutaan kirjoittaa ohjelma, joka lukee käyttäjältä 30 lämpötilaa.
- Kun lämpötilat on luettu, ohjelma tulostaa luetut lämpötilat ja niiden keskiarvon.
- Ratkaisu: käytetään *listaa*.
- Lista on rakenne, johon voidaan tallentaa useita eri arvoja.
- Indeksin avulla voidaan määrätä, mitä listan alkiota käsitellään. Esimerkiksi `lampotilat[4]`.

Listan luominen ja alkioiden lisäys

- Tyhjän listan luominen:

```
lampotilat = []
```

- Alkion lisäys listan loppuun:

```
lampotilat.append(arvo)
```

- Uuden arvon sijoittaminen indeksille `i`:

```
lampotilat[i] = arvo
```

- Esimerkkejä yksittäisen alkion käyttämisestä:

```
print("5. lampotila", lampotilat[4])
summa = summa + lampotilat[4]
```

Esimerkkejä

```
lampotilat = []
```

Esimerkkejä

lampotilat

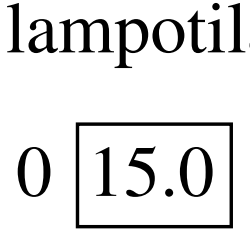
```
lampotilat = []
```

Esimerkkejä

lampotilat

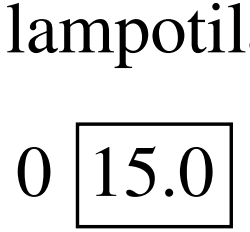
```
lampotilat = []
lampotilat.append(15.0)
```

Esimerkkejä




```
lampotilat = []
lampotilat.append(15.0)
```

Esimerkkejä




```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
lampotilat.append(15.4)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7
3	15.4

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
lampotilat.append(15.4)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7
3	15.4

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.8)
lampotilat.append(15.4)
lampotilat.append(13.8)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7
3	15.4
4	13.8

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.8)
lampotilat.append(15.4)
lampotilat.append(13.8)
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7
3	15.4
4	13.8

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
lampotilat.append(15.4)
lampotilat.append(13.8)
lampotilat[3] = 18.0
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7
3	~~15.4~~
4	13.8

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
lampotilat.append(15.4)
lampotilat.append(13.8)
lampotilat[3] = 18.0
```

Esimerkkejä

lampotilat

0	15.0
1	12.1
2	10.7
3	18.0
4	13.8

```
lampotilat = []
lampotilat.append(15.0)
lampotilat.append(12.1)
lampotilat.append(10.7)
lampotilat.append(15.4)
lampotilat.append(13.8)
lampotilat[3] = 18.0
```

Listan läpikäynti toistokäskyllä

- Listassa on `LKM` alkiota, lasketaan niiden summa:

```
summa = 0.0
i = 0
while i < LKM:
    summa += lampotilat[i]
    i += 1
```

- Läpikäynti on vielä helpompaa for-käskyn avulla:

```
summa = 0.0
for astemaara in lampotilat:
    summa += astemaara
```

Muuttuja `astemaara` saa arvokseen vuorotellen jokaisen listan alkion.

Lämpötilaesimerkki, koodi

```
def main():
    LKM = 30
    lampotilat = []
    i = 0
    print("Anna", LKM, "lampotilaa")
    while i < LKM:
        lampo = float(input("Seuraava lampotila: "))
        lampotilat.append(lampo)
        i += 1
```

Lämpötilaesimerkki, koodi jatkuu

```
    summa = 0.0
    print("Annetut lampotilat")
    for arvo in lampotilat:
        print(arvo)
        summa += arvo
    keskiarvo = summa / LKM
    print("Lampotilojen keskiarvo on", keskiarvo)


main()
```

Alkiot listaan listaa luodessa

- Luotavan listan ei tarvitse olla aluksi tyhjä:

```
numerolista = [2, 4, 6, 8]
nimilista = ['Matti Virtanen', 'Maija Maki', 'Anu Lahti']
```

- Voidaan myös luoda lista, jossa on aluksi haluttu määrä nollia.

```
LKM = 30
lampotilat = [0.0] * LKM
```

Tällöin listaan voidaan lisätä lämpötiloja sijoituskäskyllä:

```
i = 0
while i < LKM:
    lampo = float(input("Seuraava lampotila: "))
    lampotilat[i] = lampo
    i += 1
```

Listan pituus käyttäjältä

- Luotavan listan pituuden voi myös pyytää käyttäjältä.

```
lkm = int(input("Anna lampotilojen lukumaara. "))
lampotilat = [0.0] * lkm
```

- Listan pituuden saa selville Pythonissa valmiina olevalla funktiolla `len`:

```
i = 0
while i < len(lampotilat):
    lampo = float(input("Seuraava lampotila: "))
    lampotilat[i] = lampo
    i += 1
```

Välitehtävä

- Mitä seuraava ohjelma tulostaa?

```
lista = [1, 2, 3]
lista[0] = lista[1] + lista[2]
lista[2] = lista[0] + lista[1]
for alkio in lista:
    print(alkio)
```

- Vastaa sivulla http://presemo.aalto.fi/y1syksy2023

Merkkijonoja sisältävä lista

- Listan alkioiden ei tarvitse olla lukuja, vaan ne voivat olla esimerkiksi merkkijonoja.
- Saman listan eri alkiot voivat olla keskenään myös erityyppisiä.
- Listan alkiona voi olla myös toinen lista.
- Merkkijonoja sisältävät listat ovat käteviä, kun käyttäjälle halutaan tulostaa erilaisia valintavaihtoehtoja.

```
def main():
    ruokalajit = ['makaronilaatikko', 'lihapullat',\
                  'lihakeitto', 'kasvispata', 'pihvi']
    print("Valitse ruokalaji:")
    i = 0
    while i < len(ruokalajit):
        print(f"{i + 1:d}. {ruokalajit[i]:s}")
        i += 1
    valinta = int(input())
    if valinta > 0 and valinta <= len(ruokalajit):
        print("Valitsit ruuan", ruokalajit[valinta - 1])
    else:
        print("Antamallasi numerolla ei ole ruokalajia.")

main()
```

Lista parametrina ja funktion palauttamana arvona

- Lämpötiloja käsittelevää ohjelmaa voi selkiyttää jakamalla sen useampaan funktioon.
- Tiedon käsiteltävistä lämpötiloista pitää siirtyä eri funktioiden välillä.
- Tähän voidaan käyttää parametreja ja funktion paluuarvoa.
- Funktio voi palauttaa arvonaan listan ja funktiolle voidaan antaa parametrina lista.
- Jos funktio tekee muutoksia parametrina saadun listan sisältöön, näkyvät muutokset myös silloin, kun samaa listaa käytetään funktion ulkopuolella.

Esimerkki: listan palauttava funktio

```
def kysy_lampotilat():
    LKM = 30
    lampotilat = [0.0] * LKM
    i = 0
    print("Anna", LKM, "lampotilaa")
    while i < LKM:
        lampo = float(input("Seuraava lampotila: "))
        lampotilat[i] = lampo
        i += 1
    return lampotilat
```

Esimerkki: lista funktion parametrina

```
def tulosta_lampotilat(lammot):
    print("Annetut lampotilat")
    for arvo in lammot:
        print(arvo)


def laske_keskiarvo(lampotilalista):
    summa = 0.0
    for lampotila in lampotilalista:
        summa += lampotila
    lukumaara = len(lampotilalista)
    if lukumaara > 0:
        keskiarvo = summa / lukumaara
    else:
        keskiarvo = 0.0
    return keskiarvo
```

Esimerkki: pääohjelma

```
def main():
    lampolista = kysy_lampotilat()
    tulosta_lampotilat(lampolista)
    k_arvo = laske_keskiarvo(lampolista)
    print(f"Lampotilojen keskiarvo on {k_arvo:.2f}.")


main()
```