Ohjelmoinnin peruskurssi Y1

CS-A1111

8.11.2023

Oppimistavoitteet: tämän luennon jälkeen

- Tiedät lisää siitä, miten Python-ohjelmissa voi käyttää valmiita kirjastoja
- Osaat tehdä joitakin matriisien laskutoimituksia kirjaston NumPy avulla
- Tiedät tapoja käyttää pandas-kirjastoa datan käsittelyyn.
- Tämän luennon aiheet eivät kuulu kurssi- eikä tenttivaatimuksiin eikä niitä tarvitse kurssin harjoitustehtäviä tehdessä. **Kirjastoja NumPy ja pandas ei saa käyttää harjoitustehtäväratkaisuissa (A+ ei hyväksy tällaista ratkaisua)**
- Tämä luento ei ole perusteellinen opastus näiden kirjastojen käyttämiseen, vaan enemmän lyhyt esittely joistakin asioista, joita näillä kirjastoilla voi tehdä.
- Voit luennon aikana lähettää kysymyksiä ja kommentteja sivulla http://presemo.aalto.fi/y1syksy2023

Kirjasto NumPy

- Pythonin listat eivät toimi tarpeeksi tehokkaasti silloin, kun pitää käsitellä suuria datamääriä esim. tieteellisessä laskennassa.
- Kirjasto NumPy on suunniteltu tarjoamaan tehokkaat työkalut taulukoiden (sekä yksi- että moniulotteisten) käsittelyyn ja laskutoimituksiin.
- Tällä luennolla esitetään esimerkkeinä kaksiulotteisia taulukoita, joilla voi esittää matriiseja, mutta useampiulotteiset taulukot ovat yhtä mahdollisia.

Kirjaston NumPy käyttöönotto

- Kirjasto NumPy ei tule Python-tulkin asennuksen mukaisesti automaattisesti, vaan se pitää asentaa erikseen. Sivulla https://www.jetbrains.com/help/pycharm/installing-uninstalling-and-upgrading-packages.html on asennusohjeet PyCharmiin,
- Ohjelmatiedoston alussa kirjasto pitää ottaa käyttöön `import`-käskyllä. Yleinen tapa on antaa samalla kirjaston nimelle alias `np`.

```
import numpy as np
```

- Näin kirjaston funktioita ym. voi kutsua käyttämällä `np`-etuliitettä, esim.

```
matriisi1 = np.array([[1,2],[3,4]])
```

Taulukko NumPyssa

- NumPyn keskeinen tyyppi on `ndarray`, jonka avulla voidaan esittää moniulotteisia taulukoita. Toisin kuin Pythonin listalla, NumPyn taulukolla on kiinteä koko (alkioiden määrä) ja kaikilla alkioilla sama tyyppi.
- Taulukon voi luoda esimerkiksi seuraavasti (matriisi, jossa on kaksi riviä ja kaksi saraketta):

```
matriisi1 = np.array([[1,2],[3,4]])
```

Aritmetiikkaa matriiseilla

- NumPyssa voi matriisien välillä käyttää tavallisia aritmeettisia operaatioita, mutta silloin laskutoimitukset tehdään aina alkioittain

```
matriisi1 = np.array([[1,2],[3,4]])
matriisi2 = np.array([[5,6],[7,8]])
summamatriisi = matriisi1 + matriisi2
print(summamatriisi)
tulosmatriisi = matriisi1 * matriisi2
print(tulosmatriisi)
```

- Tulostus

```
[[ 6  8]
 [10 12]]
[[ 5 12]
 [21 32]]
```

Matriisien kertolasku

- Matriisien kertolasku voidaan tehdä operaattorilla `@` tai kutsumalla funktiota `dot`

```
import numpy as np

matriisi1 = np.array([[1,2],[3,4]])
matriisi2 = np.array([[5,6],[7,8]])
tulomatriisi1 = matriisi1 @ matriisi2
tulomatriisi2 = np.dot(matriisi1, matriisi2)
print(tulomatriisi1)
print(tulomatriisi2)
```

- Tulostus

```
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
```

Käänteismatriisi

- NumPy-kirjastoon kuuluvasta alimoduulissa `numpy.linalg` on erilaisia lineaarialgebrassa käytettäviä operaatioita. Alla on esimerkkinä matriisin käänteismatriisin laskeminen funktiolla `inv`.

```
import numpy as np
import numpy.linalg

mat1 = np.array([[1, 3, -2], [0, 2, 4], [2, 1, 1]])
kaanteismatriisi = np.linalg.inv(mat1)
print(kaanteismatriisi)
```

- Tulostus

```
[[-0.06666667 -0.16666667  0.53333333]
 [ 0.26666667  0.16666667 -0.13333333]
 [-0.13333333  0.16666667  0.06666667]]
```

Lineaarisen yhtälöryhmän ratkaisu NumPyn avulla

- Halutaan ratkaista yhtälöryhmä

$$\begin{cases} x + 3y - 2z = 18 \\ 2y + 4z = -2 \\ 2x + y + z = 11 \end{cases}$$

- Tämä voidaan kirjoittaa matriisiyhtälönä

$$\begin{bmatrix} 1 & 3 & -2 \\ 0 & 2 & 4 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 18 \\ -2 \\ 11 \end{bmatrix}$$

Lineaarisen yhtälöryhmän ratkaisu NumPyn avulla

- Edellisen kalvon kaltainen matriisiyhtälö (ja samalla alkuperäinen yhtälöryhmä) voidaan ratkaista käyttämällä funktiota `solve`:

```
import numpy as np
import numpy.linalg

mat1 = np.array([[1, 3, -2], [0, 2, 4], [2, 1, 1]])
vect = np.array([18, -2, 11])
solution = np.linalg.solve(mat1, vect)
print(solution)
```

- Tulostus

```
[ 5.  3. -2.]
```

- Tulostus esittää ratkaisun $x = 5, y = 3, z = -2$

Taulukon tallentaminen tiedostoon

- NumPy tarjoaa myös mahdollisuuden tallentaa koko taulukko (voi sisältää tuhansia tai miljoonia alkioita) binääritiedostoon yhdellä käskyllä ja lukea näin tallennettu taulukko tiedostosta yhdellä käskyllä.
- Tiedoston päätteen pitää olla `.npy`.
- Tiedostoon tallentaminen tapahtuu funktiolla `save`. Alla oleva ohjelma laskee kahden matriisin matriisitulon ja tallentaa tulosmatriisin tiedostoon `tulos.npy`.
- Ohjelmasta on jätetty virheenkäsittely pois.

```
import numpy as np

matriisi1 = np.array([[1,2],[3,4]])
matriisi2 = np.array([[5,6],[7,8]])
np.save("tulos.npy", matriisi1 @ matriisi2)
```

Taulukon lukeminen tiedostosta

- Funktiolla `save` tiedostoon tallennetun taulukon voi lukea funkiolla `load`. Alla oleva ohjelma lukee toisen ohjelman tiedostoon `tulos.npy` tallentaman matriisin ja tulostaa sen kuvaruudulle.

```
import numpy as np

matriisi = np.load("tulos.npy")
print(matriisi)
```

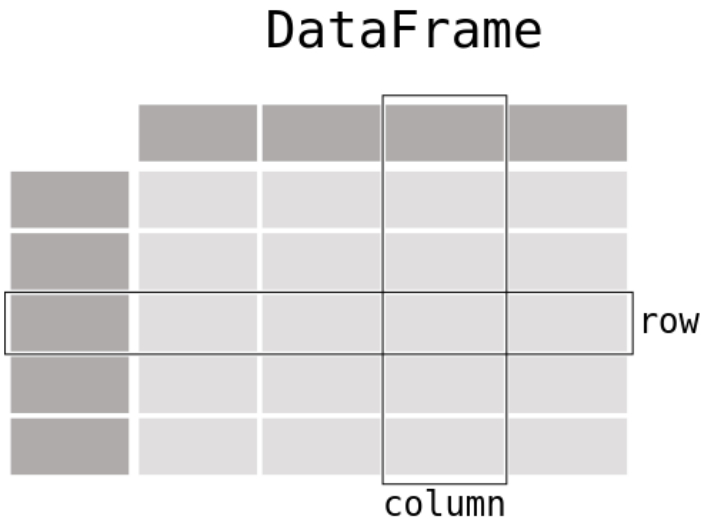
Kirjasto pandas

- Kirjaston pandas avulla on helppo käsitellä ja analysoida kaksiulotteisiin taulukoihin tallennettua dataa. Sen avulla on helppo lukea ja muokata esim Excel- ja csv-tiedostoja.
- Pandas ei tule automaattisesti Python-tulkin mukana, vaan se pitää asentaa erikseen samalla tavalla kuin NumPy.
- Ohjelmatiedoston alkuun on myös kirjoitettava `import`-käsky:

```
import pandas as pd
```

Pandas DataFrame

- Pandas-kirjastossa perustietorakenne on kaksiulotteinen taulukko, `DataFrame`.
- Taulukon sarakkeilla on otsikot ja myös rivit voi nimetä.
- Samassa sarakkeessa kaikkien arvojen on oltava samaa tyyppiä, mutta eri sarakkeissa voi olla keskenään erityyppisiä arvoja.
- Kukin `DataFramen` sarake on tyyppiä `Series`.



- Kuva on sivulta https://pandas.pydata.org/docs/getting_started/intro_tutorials/01_table_oriented.html

DataFramen luominen

- Alla oleva ohjelma luo `DataFramen` sanakirjan perusteella ja tulostaa sen.
- Taulukon voisi tulostaa myös käskyllä `print(tulokset)`, mutta jos taulukossa on paljon rivejä, tämä käsky tulostaa niistä vain osan.

```
import pandas as pd

suoritustiedot = {
  "nimi" : ["Aku Ankka", "Mikki Hiiri", "Minni Hiiri"],
  "harjoitusarvosana" : [1, 4, 5],
  "tenttiarvosana" : [2, 5, 4]
}
tulokset = pd.DataFrame(suoritustiedot)
print(tulokset.to_string())
```

DataFramen luominen, jatkuu

- Edellisen kalvon ohjelman tulostus:

```
              nimi  harjoitusarvosana  tenttiarvosana
0     Aku Ankka                    1                2
1  Mikki Hiiri                     4                5
2  Minni Hiiri                     5                4
```

Sarakkeen lisääminen

- Alla oleva ohjelma lisää taulukkoon sarakkeet `kokonaisarvosana` ja `kiitettava`, joihin tulevat arvot päätellään saman rivin muiden sarakkeiden arvojen perusteella.

```
import pandas as pd

suoritustiedot = {
  "nimi" : ["Aku Ankka", "Mikki Hiiri", "Minni Hiiri"],
  "harjoitusarvosana" : [1, 4, 5],
  "tenttiarvosana" : [2, 5, 4]
}
tulokset = pd.DataFrame(suoritustiedot)

tulokset["kokonaisarvosana"] = (tulokset["harjoitusarvosana"] +
                                tulokset["tenttiarvosana"] + 1) // 2
tulokset["kiitettava"] = tulokset["kokonaisarvosana"] >= 4
print(tulokset.to_string())
```

Sarakkeen lisääminen, jatkoa

- Edellisen kalvon ohjelman tulostus

	nimi	harjoitusarvosana	tenttiarvosana	kokonaisarvosana	kiitettava
0	Aku Ankka	1	2	2	False
1	Mikki Hiiri	4	5	5	True
2	Minni Hiiri	5	4	5	True

Avainsarakkeen lisäys

- Pandas antaa oletusarvoisesti riveille avaimeksi juoksevan numeron. Alla olevassa ohjelmassa riveille on vaihdettu avaimiksi opiskelijanumerot. Viimeisellä rivillä on haettu ja tulostettu määrättyyn avaimeen liittyvän opiskelijan tiedot.

```
import pandas as pd

suoritustiedot = {
  "nimi" : ["Aku Ankka", "Mikki Hiiri", "Minni Hiiri"],
  "harjoitusarvosana" : [1, 4, 5],
  "tenttiarvosana" : [2, 5, 4]
}
tulokset = pd.DataFrame(suoritustiedot,
                        index=["313", "112233","334455"])
print(tulokset.to_string())
print("Opiskelijan 112233 tiedot:")
print(tulokset.loc["112233"])
```

Sarakkeiden valinta

- Ohjelmassa voidaan valita, mitä sarakkeita käsitellään tai tulostetaan. Alla oleva ohjelma tulostaa taulukosta avainten lisäksi vain kahden sarakkeen tiedot.

```
import pandas as pd

suoritustiedot = {
  "nimi" : ["Aku Ankka", "Mikki Hiiri", "Minni Hiiri"],
  "harjoitusarvosana" : [1, 4, 5],
  "tenttiarvosana" : [2, 5, 4]
}
tulokset = pd.DataFrame(suoritustiedot,
                        index=["313", "112233","334455"])
print(tulokset[["nimi","tenttiarvosana"]])
```

Sarakkeiden valinta, jatkoa

- Edellisen kalvon ohjelman tulostus

```
                  nimi  tenttiarvosana
313          Aku Ankka               2
112233  Mikki Hiiri                  5
334455  Minni Hiiri                  4
```

Tilastotietojen laskeminen

- Pandas tarjoaa valmiita funkioita, joiden avulla voidaan laskea sarakkeiden arvoista erilaisia tilastotietoja.

```
import pandas as pd

suoritustiedot = {
  "nimi" : ["Aku Ankka", "Mikki Hiiri", "Minni Hiiri"],
  "harjoitusarvosana" : [1, 4, 5],
  "tenttiarvosana" : [2, 5, 4]
}
tulokset = pd.DataFrame(suoritustiedot,
                        index=["313", "112233","334455"])
print("Tenttiarvosanojen keskiarvo:",
      tulokset["tenttiarvosana"].mean())
print("Harjoitusarvosanojen minimi:",
      tulokset["harjoitusarvosana"].min())
```

Taulukon lukeminen tiedostosta

- Pandas-kirjaston avulla voi lukea yhdellä käskyllä moniin eri tiedostoformaatteihin (esim. csv, excel, json) tallennettuja taulukoita DataFrameiksi.
- Seuraavan kalvon ohjelma lukee csv-tiedostoon arvosanat.csv tallennetun taulukon niin, että rivin ensimmäisessä osassa olevaa tietoa käytetään taulukon rivin avaimena.
- Pandas käsittelee ensimmäisen sarakkeen arvoja ensin kokonaislukuina, mutta seuraavalla rivillä tämän sarakkeen arvojen tyypiksi on vaihdettu merkkijono `str`.
- Lopuksi ohjelma tulostaa koko taulukon ja sitten avaimeen "313" liittyvän rivin tiedot.

Taulukon lukeminen tiedostosta, esimerkkiohjelma

```
import pandas as pd

tulokset = pd.read_csv('arvosanat.csv', index_col=0)
tulokset.index = tulokset.index.map(str)
print(tulokset.to_string())
print(tulokset.loc["313"])
```

Taulukon tallentaminen tiedostoon

- Taulukon (`DataFrame`-olion) voi myös tallentaa tiedostoon yhdellä käskyllä. Käytettävissä on useita eri tiedostoformaatteja.
- Seuraavan kalvon esimerkkiohjelma tallentaa taulukon csv-tiedostoon.
- Ilmauksella `index=False` määrätään, että tallennettaviin tietoihin ei tule mukaan rivin avaimena käytettävää numerokenttää.

Taulukon tallentaminen tiedostoon, esimerkkiohjelma

```
import pandas as pd
suoritustiedot = {
  "nimi" : ["Aku Ankka", "Mikki Hiiri", "Minni Hiiri"],
  "harjoitusarvosana" : [1, 4, 5],
  "tenttiarvosana" : [2, 5, 4]
}
tulokset = pd.DataFrame(suoritustiedot)

tulokset["kokonaisarvosana"] = (tulokset["harjoitusarvosana"] +
                                tulokset["tenttiarvosana"] + 1) // 2
tulokset.to_csv("tulokset3.csv", index=False)
```

Datan puhdistaminen

- Käsiteltävä data on usein virheellistä tai puutteellista.
- Riviltä voi esim. puuttua täysin jonkin sarakkeen arvo tai sarakkessa voi olla arvo, joka on selvästi virheellinen (esim. lämpötila -300 C)
- Pandas tarjoaa keinot muokata automaattisesti puutteellisia tai virheellisiä arvoja, esim. korvata puuttuvat tai virheelliset arvot oletusarvolla tai poistaa rivit, joissa on selvästi puutteellisia arvoja.
- Seuraavan kalvon esimerkkiohjelma lukee tiedostoon `data.csv` tallennetun taulukon. Se korvaa sarakkeesta result puuttuvat arvot nollalla ja poistaan kokonaan ne rivit, joissa puuttuu jonkin muun sarakkeen arvo.

Datan puhdistaminen, esimerkki

```
import pandas as pd

tutkimusdata = pd.read_csv('data.csv')
print("Alkuperainen data")
print(tutkimusdata.to_string())
tutkimusdata["result"].fillna(0, inplace = True)
print("Tulosten taydentamisen jalkeen:")
print(tutkimusdata.to_string())
siistitty = tutkimusdata.dropna()
print("Poistettu rivit, joista muun sarakkeen arvo puuttuu:")
print(siistitty.to_string())
```

Lisätietoja

- Näillä kalvoilla on käsitelty hyvin lyhyesti vain hyvin pientä osaa kirjastojen NumPy ja pandas tarjoamista mahdollisuuksista.
- Ohjelmista on jätetty usein tarvittava virheenkäsittely pois.
- Aalto Scientific Computing -ryhmän ylläpitämillä sivuilla `https://aaltoscicomp.github.io/python-for-scicomp/` on kurssimateriaalia näiden kirjastojen käytöstä. Siitä on hyvä aloittaa lisätiedon haku.
- Kirjastojen omilla sivuilla `numpy.org` ja `pandas.pydata.org` on perusteellisimpia dokumentteja.