Python harjoitukset

Harjoitukset

1.	Tilastot	- Kirjastojen käyttö (Math) ja python syntaksin kertaus.
2.	Datan hakeminen REST rajapinnasta	- JSON datan käsittely.
3.	Tiedoston käsittely	- Tiedostojen käsittely, luku ja kirjoitus.
4.	Komentoriviparametri	- Tiedostojen ajo komentoriviltä ja tiedon syöttö ajettaessa.
5.	CRON job	- Ohjelmien ajastus ja automatisointi.
6.	Health check	- Palvelimen tilan valvonta.
7.	★ Emailin lähetys	- Hälyytykset ongelmatilanteissa.
8.	★ Automaattinen ilmoitus (tee ensin 6, 7, 8)	- Valvonnan automatisointi.
9.	★ Sieve of Eratosthenes	- Algoritmien rakentaminen.

★ Merkityt ovat lisätehtäviä

Harjoitus 1: Tilastot

- Käytä ulkopuolista kirjastoa apuna laskennassa, esim Math.
- Tee ohjelma joka:

1. Ottaa käyttäjältä yhden rivin syötettä.
   - Syöte koostuu pilkulla erotetuista numeroista.
2. Laskee ja tulostaa numeroista:
   1. pienimmän (min),
   2. suurimman (max),
   3. keskiarvon (mean),
   4. mediaanin (median), sekä
   5. moodin (mode).

Harjoitus 2: Datan hakeminen REST rajapinnasta

- Tee ohjelma, joka hakee JSON dataa requests kirjaston avulla.
- Tee ohjelma, joka hakee Python repositoryja Githubin rajapinnasta
  1. Hae JSON-dataa osoitteesta https://api.github.com/search/repositories?q=language:python (voit ensin kurkata vaikka selaimella millaista dataa osoitteesta löytyy)
  2. Tulosta data rivi kerrallaan, seuraavassa muodossa:
     {forks}. {name}: {description}
  3. ★ Järjestä tuloksena saadut vastaukset ”forks” –arvon mukaan laskevassa järjestyksessä.

- Linkkejä
  - Requests kirjaston käyttö https://docs.python-requests.org/en/latest/user/quickstart/#make-a-request
  - JSON syntaksi https://www.w3schools.com/js/js_json_syntax.asp
  - Järjestämistä https://www.geeksforgeeks.org/python-sort-json-by-value/

Harjoitus 3: Tiedoston käsittely

- Tee ohjelma joka, lukee sanoja tiedostosta rivi kerrallaan.
  1. Jokainen sana on omalla rivillään, joten 1 rivi == 1 sana.
  2. Järjestää sanat järjestykseen ensisijaisesti pituuden mukaan
  3. Kirjoittaa järjestetyt sanat toiseen tiedostoon
  4. Ohjelma ei saa kaatua vaikka tiedostonkäsittelyssä olisi virheitä (try/except)

Esim ”kuu, vii, nee, koo, kaa, yy” järjestettäisiin seuraavanlaisesti:
-> yy, kaa, koo, kuu, nee, vii
(lyhin ensimmäisenä ja yhtä pitkät akkosjärjestyksessä)

Harjoitus 4: Komentoriviparametri (Command Line Argument)

- Tee ohjelma, jolle voi antaa komentoriviparametrin
  - Ohjelman tulee tehdä jotakin annetulla parametrillä. Alla ideoita, mutta voit myös keksiä oman.
    - Voit käyttää esim Harjoituksen 3 koodia ja antaa tiedoston nimen komentoriviparametrinä
    - Voit toteuttaa laskimen ja antaa luvut, sekä operaation parametreinä
    - Tee tutorial https://docs.python.org/3.3/howto/argparse.html#id1

- Näistä löytyy apua
  - https://docs.python.org/3.3/library/argparse.html
  - https://www.tutorialspoint.com/python/python_command_line_arguments.htm
  - https://realpython.com/python-command-line-arguments/
  - https://www.geeksforgeeks.org/command-line-arguments-in-python/

CRON job & CRON tab

- CRON job on ajastettu tapahtuma, jonka avulla voidaan automatisoida ohjelman suoritus. CRON job käyttää omaa syntaksia toistuvuuden määrittelemiseen (esim suorita tiedosto joka päivä tiettyyn kellonaikaan).
  - Kurkkaa syntaksi täältä https://crontab.guru/
- CRON tab on tiedosto, johon CRON jobit tallennetaan. Perjaatteessa tämä on siis lista, jossa kerrotaan mikä tapahtuma pitää suorittaa mihinkin aikaan.
  - Kurkkaa esimerkki CRON tabiin kirjoitettavasta rivistä https://tecadmin.net/crontab-in-linux-with-20-examples-of-cron-schedule/

Harjoitus 5: CRON job

1. Toteuta CRON job linux ympäristössä. Suoritettava taski voi olla mitä vain, mutta taskin suoritus on helpoin todentaa jos siitä jää jälki. Tee esim python sovellus joka kirjoittaa kellonajan paikalliseen tiedostoon.
   - Ohjeita löytyy täältä: https://stackabuse.com/scheduling-jobs-with-python-crontab/ (Tässä esimerkissä ei itse tarvitse kirjoittaa mitään CRON tabiin, sillä siinä käytetään python-crontab kirjastoa tämän tekemiseen.)
   - Vinkki: Voit koodata VSCodella ja siirtää tiedoston virtuaalikoneeseen esim git pull komennolla, blob storagesta tai Puttyn avulla
2. ★ Lisätehtävä: Toteuta CRON job Windows ympäristössä.
   - https://datatofish.com/python-script-windows-scheduler/

- Linkkejä
  - https://en.wikipedia.org/wiki/Cron
  - https://towardsdatascience.com/automate-your-python-scripts-with-task-scheduler-661d0a40b279

Harjoitus 6: Health Check

! Health checkien lähettäminen palvelimelle on yleinen käytäntö, jolla varmistetaan palvelimen tila (esim nginx). Normaali tilanteessa HTTP status koodi “200 OK” on vastaus, jonka terve palvelin palauttaa.

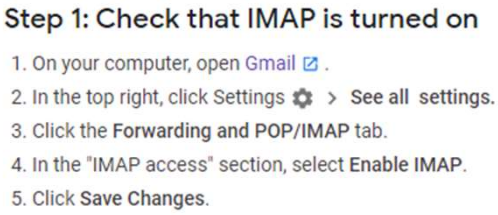
- Tee ohjelma, joka lähettää health checkin palvelimelle.
  1. Käynnistä Azureen virtuaalikone “Debian 10 buster”, jossa Apache palvelin. Startup Scripti (cloud-init) löytyy alta.
  2. Muokkaa tarvittaessa Firewall asetuksia siten, että se sallii liikenteen porttiin 80 ainakin omalta koneeltasi.
  3. Voit kokeilla, että yhteys toimii käyttämällä curlia tai kurkkaamalla selaimesta (osoite 4 kohdassa ja yksi esimerkki linkki alla).
  4. Tee nyt python ohjelma joka lähettää ajettaessa HTTP get pyynnön osoitteeseen http://my-external-ip/health.html (korvaa “my-external-ip“oman palvelimen ip osoitteella).
  5. Varmista, että ohjelma toimii ja get pyynnön HTTP status on 200 OK, selvitä mistä johtuu jos tämä ei onnistu.
  6. Poista lopuksi virtuaalikoneelta tiedosto /var/www/html/health.html ja aja python ohjelma uudestaan. Nyt HTTP statuksen pitäisi olla 404 Page not found. Tällä poistamisella simuloidaan sitä, että palvelin olisi kaatunut. Silloin Health checkin ei kuulukaan mennä läpi.

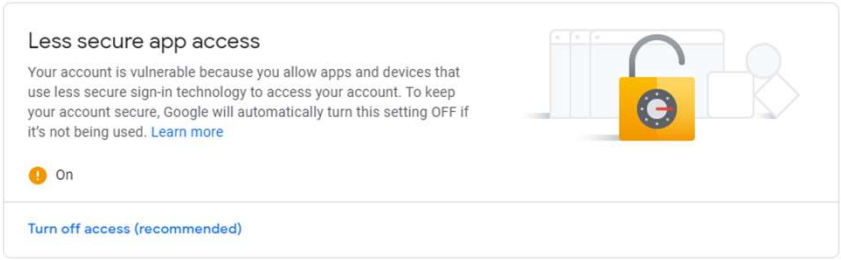
```
#!/bin/bash
sudo apt update && sudo apt -y install apache2
echo '<!doctype html><html><body><h1>Hello World!</h1></body></html>' | sudo tee /var/www/html/index.html
echo '<!doctype html><html><body><h1>Healthy</h1></body></html>' | sudo tee /var/www/html/health.html
```

- Linkkejä aiheesta
  - Miten curlataan https://curl.se/docs/manpage.html
  - Healthy check https://www.consul.io/api/health.html

Harjoitus 7: Emailin lähetys - lisätehtävä

- Tee sovellus joka lähettää sähköpostia. (Toimii ainakin gmail tilillä, kun Less secure app access on päällä ja IMAP on käytössä.)
  - Tietoturvasyistä älä käytä omaa mailia vaan tee uusi tai käytä jotakin roskapostia
  - Ohjeet: https://en.wikibooks.org/wiki/Python_Programming/Email
    - Sending mail kohdan alla oleva riittää





Harjoitus 8: Automaattinen ilmoitus

- Tässä tehtävässä hyödynnetään kolmen aiemman tehtävän toiminnallisuutta (6, 7, 8)
- Tee sovellus joka lähettää sähköpostiisi ilmoituksen, mikäli automatisoitu health check ei onnistu.
- Vaatimukset:
  1. Toteuta python ohjelma, joka tekee palvelimelle health checkin (tehtävä 7).
  2. Ajasta Health check CRON jobia (tehtävä 6) käyttäen siten, että se lähettää palvelimelle pyynnön 15sec välein.
  3. Jos Health check on jotain muuta, kuin OK, lähetä sähöpostiisi (tehtävä 8) tästä huomautus.

Harjoitus 9: Sieve of Eratosthenes - Lisätehtävä

- Toteuta Sieve of Eratosthenes
  1. Ohjelma pyytää käyttäjältä numeron
  2. Ohjelma tulostaa kaikki olemassa olevat alkuluvut annettuun numeroon saakka, välilyönnillä eroteltuna
  3. Alkulukujen etsimiseen käytetään algoritmia nimeltä ”Sieve of Eratosthenes”
  4. Algoritmin speksit löytyvät osoitteesta: https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
  5. Esimerkkiajo:

     10

     2 3 5 7


AW
Academy