CT10A0013 Ohjelmointi Pythonilla

L08: Uudelleenkäyttö ja harjoitustyö

Uolevi Nikula

Päivän asiat

- Teoria
  - Uudelleenkäyttö
  - Aliohjelmat
  - Kirjastot
- Käytäntö
  - Kirjaston käyttöönotto
  - Ulkoisen kirjaston lisääminen Pythoniin
- Koodiesimerkkejä
  - Kirjaston peruskäyttö
  - datetime-kirjaston käyttö
  - Itse tehty kirjastomoduuli ja sen käyttö
- Lopuksi

Teoria

Uudelleenkäyttö
Aliohjelmat
Kirjastot

Ohjelmia

- Tähän mennessä olemme oppineet tekemään ohjelmia
  - Pääosin pieniä 1-20 rivin ohjelmia
  - Valikkopohjaiset ohjelmat **aliohjelmilla** ovat isompia, kymmeniä tai satoja rivejä
  - Ison ohjelman tekeminen teettää töitä
- Tähän asti aliohjelmia on käytetty samassa tiedostossa olevien ohjelmien kanssa
- Ohjelmien koon kasvaessa herää kysymys työn tehostamista
  - Voisiko muiden tekemiä (ali)ohjelmia hyödyntää jotenkin omissa ohjelmissa?
  - Miten aiemmin tehtyjä (ali)ohjelmia voisi hyödyntää mahdollisimman tehokkaasti?

Uudelleenkäyttö

- Ohjelmointiympäristön mukana tulee valmiita aliohjelmia ja niitä voi hankkia myös muista lähteistä
- Valmiit aliohjelmakirjastot ovat keskeisessä roolissa tehostettaessa ohjelmointityötä
- Tällä luennolla tutustumme aliohjelmakirjastoihin, niiden hyötyihin ja hyödyntämiseen sekä oman aliohjelmakirjaston tekemiseen
- Valmiiden ohjelmien hyödyntämistä uusien ohjelmien teossa kutsutaan **uudelleenkäytöksi**
- **Suunnittelu ja uudelleenkäyttö** DI 2. vsk syventävällä ohjelmistokehityskurssilla
  - Tyypillisesti tiimit suunnittelivat arkkitehtuuria → n. 4500 rivin ohjelmia
  - ”Tämä on vain koodaamista, ei tarvita kehikkoja tms.” → n. 9000 rivin ohjelma
    - Ts. ohjelmaa ei suunniteltu kunnolla
  - Yksi tiimi käytti sisällönhallintajärjestelmää (Drupal) → n. 1500 rivin ohjelma
    - Ts. uudelleenkäyttö

Ohjelmien perusrakenteista

- Tietokoneohjelmissa toistuvat samat ohjelmoinnin perusrakenteet
  - Esim. muuttujat, toistorakenteet, valintarakenteet, loogiset operaatiot, syöttö ja tulostus, tiedostot
- Ohjelmointi on pitkälti näiden perusrakenteiden ja toimintojen soveltamista kulloinkin käsillä olevan ongelman ratkaisemiseksi
  - Isossa ohjelmassa näitä samoja rakenteita on paljon
  - Samojen ratkaisujen kirjoittaminen jatkuvasti uudestaan on työlästä ja usein turhauttavaa
  - Siksi **samoihin asioihin** keskittyvät ohjelmat/koodi kannattaa ratkaista yhden kerran kunnolla ja käyttää jatkossa aina samaa ratkaisua

Ohjelmia – yleisiä ja spesifisiä

- Tähän mennessä tehdyt ohjelmat ovat olleet usein luonteeltaan yleisiä ohjelmointitehtäviä ja perusasioiden harjoittelua
- Jokaisen ohjelman käyttöympäristössä eli sovellusalueella on tehtäviä, jotka ovat tyypillisiä ko. sovellusalueelle, esim.
  - Pankit: valuuttamuunnokset, koronlaskenta
  - Energiatekniikka: veden ominaisuudet (paine, lämpötila, lämpökapasiteetti, kosteuspitoisuus)
  - Konetekniikka: pituudet, pinta-alat, yksikkömuunnokset, aineen ominaisuudet kuten lujuudet jne.
  - Tietokoneavusteinen suunnittelu: koordinaatistot, värit, trigonometria
  - Tietoturva: salausalgoritmit

Valmiiden ratkaisujen hyväksikäyttö

- Ohjelmien monipuolistuessa niiden sisältämän toiminnallisuuden määrä kasvaa nopeasti
- Usein keskitytään omiin erityisongelmiin, ydinosaamiseen, ja hankkimaan muu käyttövalmiina komponentteina
- Esimerkkejä käyttövalmiista komponenteista
  - Tietovarastoratkaisun lähtökohtana voi olla kaupallinen tietokantatoimittaja
  - Raporttien luomiseen voidaan käyttää raporttigeneraattoreita
  - Kaikki yliopiston kurssien palautekyselyt tehtiin yhdellä yhteisellä lomakkeella ja nyt ollaan siirtymässä Moodle-työkaluun
  - Tiedostoja voidaan katsella Adobe Acrobat Readerillä pdf-muodossa
  - Trigonometriset funktiot on toteutettu ohjelmointikielen kirjastomoduulina

Python: aliohjelmia ja kirjastoja

- Python-ohjelmointikielelle on tarjolla useita aliohjelmakirjastomoduuleja, jotka sisältävät käyttövalmiita aliohjelmia ja vakioita eri ongelma- ja tehtäväalueille, esim.
  - math: matemaattisia funktioita (pi, sin, log, …)
  - random: satunnaislukuja
  - datetime: päivämääriä ja kellonaikoja
  - sys: systeemiin liittyviä muuttujia ja aliohjelmia
  - os: käyttöjärjestelmäpalveluita (prosesseja ja tiedostonkäsittelyä)
  - Tkinter: graafinen käyttöliittymä
  - urllib2: web (http) –sivujen lukemisessa käytettäviä aliohjelmia
  - svgwrite: vektorigrafiikan piirto

Uudelleenkäytön etuja ja haittoja

- Koodin uudelleenkäyttö perustuu tyypillisesti aliohjelmakirjastoihin
- Uudelleenkäytön eli valmiiden komponenttien keskeisiä etuja
  - **Työnteko nopeutuu ja tehostuu**
  - **Laatu paranee**
- Uudelleenkäytön yleisiä ongelmia
  - Uudelleenkäyttö ei ole mahdollista, mikäli ei ole tietoa komponenttien olemassaolosta (esim. aliohjelmat, vakiot, luokat jne.)
  - Uudelleenkäyttö on vaikeaa, mikäli ei ymmärrä käytettävien komponenttien rajoitteita ja käyttötapoja kunnolla
  - Nämä ongelmat voidaan ratkaista opettelulla, joka edellyttää aikaa

Yhteenveto kirjastoista

- Aliohjelmakirjastot ovat yleinen tapa laajentaa ohjelmistojen toiminnallisuutta ja niitä käytetään sekä kaupallisissa yleisohjelmistoissa (esim. Excel ja Word) että ohjelmoinnissa (Python, C, Java, C++, …)
- Kirjastot tarjoavat käyttövalmiita ratkaisuja eri ongelma-/sovellusalueille
- Kirjasto-ohjelmien hyväksikäyttö on usein sopivien valmiiden aliohjelmien liimaamista yhteen sopivalla tavalla
- Samaan kirjastoon sijoitettavat aliohjelmat muodostavat loogisen kokonaisuuden, jotta niiden löytäminen on helpompaa, vrt. esim.
  - matematiikka (**math**): pi, pow, sin, cos, radians, log, log10, sqrt, …
  - satunnaisluvut (**random**): randint, randrange,…
  - päivämäärät ja kellonajat (**datetime**): datetime, now, strptime, strftime, timedelta, …

Kirjastojen käyttö tällä kurssilla

- Tällä kurssilla saa käyttää vain kurssimateriaaleissa esiteltyjä kirjastoja
  - Nämä kirjastot ovat käytettävissä tarvittaessa myös CodeGradessa ja EXAMissa, muut kirjastot eivät välttämättä ole käytössä
  - Näiden kirjastojen algoritmit ovat sopivia tälle kurssille
    - Suorituskyky on sopiva / riittävä
    - Tällä kurssilla harjoitellaan perusalgoritmien toteuttamista
- Muut kirjastot eivät välttämättä ole sopivia aloittelijoille ja/tai tälle kurssille ja siksi niitä ei saa käyttää

Käytäntö

Kirjaston käyttöönotto

Ulkoisen kirjaston lisääminen Pythoniin

Kirjastojen käyttöönotto

- Kirjaston sisältämät aliohjelmat, vakiot ja luokat saa käyttöön import-komennolla eli

```
import math
```

- Tämän jälkeen math-kirjaston sisältämiä funktioita, vakioita ja luokkia voidaan käyttää pistenotaatiolla, esim.

```
print(math.pi)
```

- Kunkin kirjaston sisältämät aliohjelmat, vakiot ja luokat löytyvät dokumentaatiosta (esim. IDLE | F1 | Library Reference)
- Aliohjelmat ovat itse tehtyjä aliohjelmia vastaavia, mutta ne on sijoitettu erilliseen kirjastoon
  - Kirjastoon sijoitettujen aliohjelmien käyttö edellyttää **`import`**-käskyä
  - Kirjastot ovat .py –päätteisiä kooditiedostoja

Ulkoisen kirjaston lisääminen Pythoniin

- Katso L08 asennusvideo, alla on tällä kurssilla käytettävät versionumerot
- Python-asennus – ei tarvitse tehdä uudestaan, jos kaikki toimii
  - Python versio ja linkit löytyvät Moodlen Perustiedot-lehdeltä
  - IDLE, PIP ja dokumentit mukaan, samoin launcher ja Python lisätään ympäristömuuttujiin (PATH)
- **Kirjastoja**
  - matplotlib, svgwrite, numpy, numpy sisältyy matplotlib:iin
  - Asennus onnistuu PIP:llä eli
    - pip install matplotlib==x.y.z
    - pip install svgwrite==x.y.z
    - Asennuksessa käytettävä versionumero “x.y.z” löytyy Moodlen Perustiedot-lehden lopusta
- Ulkoisen kirjaston lisäys ei ole kurssin ydinasioita ja kurssin suorittaminen ei edellytä ulkoisten kirjastojen asennusta
  - L10 katsomme numpy-matriisia esimerkkinä Pythonin ulkopuolisesta tietorakenteesta ja sen käytöstä
  - Data-analytiikkaa katsotaan tarkemmin tilanteen mukaan kurssin lopussa
- Tilantarve: Python n. 100MB, peruskirjastot n. 100MB (matplotlib+svgwrite+numpy)

Koodiesimerkkejä

Kirjaston peruskäyttö
datetime-kirjaston käyttö
Itse tehty kirjastomoduuli ja sen käyttö

Kirjaston peruskäyttö

```
import math # kirjastot tuodaan ohjelmaan tiedoston alussa

print("e toiseen on", math.exp(2))
print("Neliöjuuri 10:stä on", math.sqrt(10))
print("sin(2) radiaaneina on", math.sin(2))
print("Piin likiarvo on", math.pi)
```

datetime –kirjaston käyttö

```
>>> # Kirjaston käyttöönotto
>>> import datetime

>>> # Otetaan aika järjestelmästä, sekä päivämäärä että kello
>>> Nyt = datetime.datetime.now() # huom. kirjasto.luokka.jäsenfunktio
>>> Nyt
datetime.datetime(2010, 4, 21, 12, 43, 25, 987000)

>>> # Tulostetaan päivämäärä ja kello merkkijonoina, strftime ja muotoiltu tulostus
>>> Nyt.strftime("Tänään on %d.%m.%Y ja kello näyttää olevan %H:%M.")
'Tänään on 21.04.2010 ja kello näyttää olevan 12:43.'

>>> # Edellisen käänteisoperaatio eli merkkijonosta aika-olio strptime -funktiolla
>>> datetime.datetime.strptime("30.10.2019 12:58", "%d.%m.%Y %H:%M")
datetime.datetime(2019, 10, 30, 12, 58)

>>> # Päivämäärillä laskeminen, päiviä voi vähentää toisistaan kuten lukuja
>>> Syntymapaiva = datetime.date(1989, 5, 30)
>>> # Otetaan käyttöön vain päivä ilman kellonaikaa
>>> TamaPaiva = datetime.date.today()
>>> Ika = TamaPaiva - Syntymapaiva
>>> Ika.days
7631
```

Itse tehty kirjastomoduuli ja sen käyttö

```
#################################################################
# Tiedosto 1 eli kirjasto: omamoduli.py
def terve():
    print("Tämä tulostuu moduulin omamoduli aliohjelmasta 'terve'.")



#################################################################
# Tiedosto 2/pääohjelma: L08Luentodemo.py, käyttää yo. kirjastoa
import omamoduli
omamoduli.terve()
```

Valikkopohjainen ohjelma kirjastolla

```
# 20221028 L08Demo.py un L08 Laajeneva demo: kirjasto
import L08DemoKirjasto

def paaohjelma():
    Valinta = 1
    TiedostoLue = "L06Lue.txt"
    TiedostoKirjoita = "L08Kirjoita.txt"
    ListaSyote = []
    ListaTulos = []
    IndeksiMax = None
    Indeksi = 0
    while (Valinta != 0):
        Valinta = L08DemoKirjasto.valikko()
        if (Valinta == 1):
            if (IndeksiMax == None):
                ListaSyote = L08DemoKirjasto.lueTiedosto(TiedostoLue, ListaSyote)
                IndeksiMax = len(ListaSyote) - 1
            else:
                Indeksi += 1

            if (Indeksi <= IndeksiMax):
                Merkkijono = ListaSyote[Indeksi]
            else:
                print("Merkkijonot loppuivat, lopeta ohjelma.")
        elif (Valinta == 2):
            ListaTulos.append(Merkkijono)
            print("Lisätty listaan merkkijono '" + Merkkijono + "'.")
        elif (Valinta == 3):
            Merkit = Merkkijono[::-1]
            ListaTulos.append(Merkit)
            print("Lisätty listaan merkkijono '" + Merkit + "'.")
        elif (Valinta == 0):
            print("Lopetetaan.")
        else:
            print("Tuntematon valinta, yritä uudestaan.")
        print()
    L08DemoKirjasto.tallennaTiedosto(TiedostoKirjoita, ListaTulos)
    ListaSyote.clear()
    ListaTulos.clear()
    print("Kiitos ohjelman käytöstä.")
    return None

paaohjelma()
```

```
# 20221028 L08DemoKirjasto.py un Laajeneva demo: kirjasto

def valikko():
    print("1) Lue merkkijono")
    print("2) Lisää listaan merkkijono etuperin")
    print("3) Lisää listaan merkkijono takaperin")
    print("0) Lopeta")
    Syote = input("Anna valintasi: ")
    Valinta = int(Syote)
    return Valinta

def lueTiedosto(Tiedosto, Lista):
    Tdsto = open(Tiedosto, "r", encoding="UTF-8")
    Rivi = Tdsto.readline()[:-1]
    while (len(Rivi) > 0):
        Lista.append(Rivi)
        Rivi = Tdsto.readline()[:-1]
    Tdsto.close()
    Tulosta = "Luettu tiedosto '" + Tiedosto + "'."
    print(Tulosta)
    return Lista

def tallennaTiedosto(Tiedosto, Lista):
    Tdsto = open(Tiedosto, "w", encoding="UTF-8")
    for Str in Lista:
        Rivi = Str + '\n'
        Tdsto.write(Rivi)
    Tdsto.close()
    Tulosta = "Tallennettu tiedosto '" + Tiedosto + "'."
    print(Tulosta)
    return None
```

Lopuksi

Osaamistavoitteet

Osaamistavoitteet

- Uudelleenkäyttö
  - Mistä kysymys ja miksi tehdä
  - Hyötyjä ja haittoja
- Kirjastot ja niiden käyttö
  - Erityisesti math, random ja datetime kirjastot
    - datetime harjoitustyön tavoitetasolla
  - Oman kirjaston luominen
- Ulkoisen kirjaston lisääminen omalle koneelle

Täydennyksiä oppaan lukuun 8

Tyyliohjeita pienille Python-ohjelmille
ASPAn tarkistukset
Oppaan esimerkit ja käsitellyt asiat

Pienen Python-ohjelman tyyliohjeet

- Kirjastojen sisällytyskäskyt tulevat Python-tiedoston ensimmäisiksi päätasolla oleviksi Python-käskyiksi
- Kirjastoon liittyvät kiintoarvot ja luokat määritellään kirjastossa
  - Muissa tiedostoissa, esim. pääohjelmassa, näitä voidaan käyttää pistenotaation avulla samalla tavoin kuin aliohjelmia
  - Huom. Nämä tulee määritellä yhdessä tiedostossa ja niihin viitataan tai ne välitetään parametreina muualla tarpeen mukaan, ts. näitä ei tule määritellä eri tiedostoissa useita kertoja
- Valikkopohjaisessa ohjelmassa kannattaa laittaa
  - Pääohjelma ja valikko-aliohjelma yhteen tiedostoon
  - Muut aliohjelmat kirjastoon, esim. tiedoston luku, kirjoitus ja analyysi
- Tällä kurssilla ei saa käyttää muita kuin kurssimateriaaleissa käsiteltyjä kirjastoja

ASPAn L08 tarkastukset

- Kirjastoihin liittyen ASPA tarkastaa ohjelmasta seuraavat asiat
  - Tarvittavien kirjastojen sisällytyskäskyt on sijoitettu Python-tiedoston ensimmäisiksi päätasolla oleviksi Python-käskyiksi
  - Sisällytysten jälkeen tulee kiintoarvot jne.

Käsitellyt asiat oppaan luvussa 8

- Kirjaston käyttö: Esimerkki 8.1 ja 8.3, datetime-muotoilut Taulukko 8.1
- Oman kirjaston teko: Esimerkki 8.2 ja 8.3
- Asennus eli ulkoisen kirjaston lisääminen Pythoniin
- Erilaisia kirjastoja: math, random, datetime, time, urllib, fractions

- **Huom. Kokoava esimerkki 8.3, datetime-kirjaston ja oman kirjaston käyttö, luokat, tiedoston luku, datan analyysi**