Ohjelmoinnin perusteet Y Python

T-106.1208

28.3.2011

Graafiset käyttöliittymät

- Tähän asti kirjoitetuissa ohjelmissa on ollut tekstipohjainen käyttöliittymä.
- Useimmissa nykyisissä ohjelmissa on kuitenkin graafinen käyttöliittymä, jossa käyttäjä ohjaa ohjelman toimintaa erilaisten komponenttien, kuten valikoiden, tekstikenttien, painikkeiden jne avulla.
- Tällä luennolla käydään läpi joitakin perusasioita graafisen käyttöliittymän kirjoittamisesta.
- Lisää perustietoa suomeksi on saatavissa esimerkiksi oppaasta `www.it.lut.fi/publications/files/publications/490/Python-Tkinteropas_LTY2007.pdf`
- Tarkempaa tietoa englanniksi on saatavilla sivulta `http://www.pythonware.com/library/tkinter/introduction/`

Yleistä graafisen käyttöliittymän kirjoittamisesta

- Graafisen käyttöliittymän kirjoittamiseen tarvitaan sopiva moduuli. Vaihtoehtoisia moduuleja on tarjolla useita, mutta niistä yleisemmin käytetty on `Tkinter`, jonka on yleensä asennettu Python-ohjelmointiympäristön mukana.
- Graafisen käyttöliittymän määrittelevän tiedoston alkuun kirjoitetaan siis

  ```
  import Tkinter
  ```

- Yleensä graafisen käyttöliittymän määritettelevä koodi kirjoitetaan luokan sisään. Käyttöliittymän ikkunaa luodessa tehtävät toimenpiteet kirjoitetaan metodin `__init__` sisään.
- Ikkuna luodaan ja ohjelma käynnistetään luomalla olio käyttöliittymän määrittelevästä luokasta.

Esimerkki 1: tyhjä ikkuna

```
import Tkinter

class Ikkuna:
    def __init__(self):
        self.paaikkuna = Tkinter.Tk()
        self.paaikkuna.title("Esimerkki 1")
        Tkinter.mainloop()


esimerkki_ikkuna = Ikkuna()
```

Vaihtoehto esimerkille 1

- Vähänkin suuremmissa ohjelmissa tarvitaan useita eri asioita `Tkinter`-moduulista. Jos `import`-käsky kirjoitetaan vähän toisella tavalla, ei moduulin nimeä tarvitse kirjoittaa aina moduulista tuotujen luokkien ja metodien nimien eteen.

```
from Tkinter import *

class Ikkuna:
    def __init__(self):
        self.paaikkuna = Tk()
        self.paaikkuna.title("Esimerkki 1")
        mainloop()


esimerkki_ikkuna = Ikkuna()
```

Komponentteja ikkunaan

- Moduulissa `Tkinter` on eri luokkia erilaisten komponenttien (engl. widget) luomista varten, esimerkiksi painikkeita varten luokka `Button` ja tekstin esittämistä varten luokka `Label`.
- Kun halutaan lisätä ikkunaan joku komponentti, luodaan vastaavan luokan olio. Oliota luodessa kerrotaan, mihin komponentti tulee (esim. pääikkuna).
- Lisäksi komponentteja luodessa voidaan antaa muita tietoja, kuten komponenttiin tuleva teksti.
- Komponentin luominen ei vielä lisää komponenttia ikkunaan, vaan tämä pitää tehdä erikseen metodilla `pack`.
- Seuraavan kalvon esimerkkiohjelmassa luodaan ikkuna, jossa on yksi teksti.

Komponentteja ikkunaan, koodi

```
from Tkinter import *

class Teksti_ikkuna:
    def __init__(self):
        self.paaikkuna = Tk()
        self.paaikkuna.title("Esimerkki 2")
        self.teksti = Label(self.paaikkuna, \
                            text = "   Tekstia ikkunassa!  ")
        self.teksti.pack()
        mainloop()


oma_ikkuna = Teksti_ikkuna()
```

Painike ikkunaan

- Ikkunaan voidaan luoda painikkeita luokan `Button` avulla.
- Pelkkä painikkeen luominen ja sen liittäminen ikkunaan ei riitä. Pitää myös kertoa, mitä ohjelman halutaan tekevän, kun painiketta on painettu.
- Ikkunan määrittelevään luokkaan kirjoitetaan oma tapahtumankäsittelijämetodi, joka kertoo, mitä ohjelman halutaan tekevän painiketta painettaessa.
- Painiketta luodessa metodin nimi annetaan luokan `Button` konstruktorille `command`-nimisenä parametrina.
- Kun painiketta on painettu, Python-tulkki kutsuu automaattisesti sille määriteltyä tapahtumankäsittelijämetodia.

Dialogin käyttäminen

- Painike-esimerkkiohjelmassa halutaan, että painikkeen painaminen saa aikaan sen, että avataan uusi dialogi-ikkuna.
- Moduulissa `tkMessageBox` on valmiita funktioita, joiden avulla voi luoda dialogi-ikkunoita, joissa on haluttu otsikko ja teksti.
- Esimerkkiohjelmassa on käytetty moduulin metodia `showinfo`, joka luo tiedoteikkunan.

Painike-esimerkki, koodi

```
from Tkinter import *
import tkMessageBox

class Painikeikkuna:
    def __init__(self):
        self.paaikkuna = Tk()
        self.paaikkuna.title("Esimerkki 3")
        self.nappi = Button(self.paaikkuna, \
                            text = "Paina minua!", \
                            command = self.anna_ilmoitus)
        self.nappi.pack()
        mainloop()
```

Painike-esimerkki, koodi jatkuu

```
    def anna_ilmoitus(self):
        tkMessageBox.showinfo("Vastaus", \
                              "Hienoa, osasit toimia oikein")


oma_ikkuna = Painikeikkuna()
```

Syötteen lukeminen käyttäjältä

- Seuraavassa esimerkissä kirjoitetaan ohjelma, joka muuntaa käyttäjän fahrenheiteina antaman lämpötilan celsius-asteiksi.
- Tarvitsemme komponentin, jonka avulla käyttäjä voi antaa haluamansa lämpötilan.
- Tähän voi käyttää tekstikenttää, joka luodaan luokan `Entry` avulla.
- Kenttään kirjoitettu teksti voidaan lukea metodin `get` avulla.
- Metodi `get` palauttaa tekstin merkkijonona. Se pitää muuntaa desimaaliluvuksi laskemista varten.
- Käyttäjä ilmoittaa painiketta painamalla siitä, että hän on jo antanut luvun. Muunnoksen tekevä ohjelman osa on siis kirjoitettu painikkeen tapahtumankäsittelijämetodin sisään.
- Esimerkkiohjelmassa muunnoksen tulos ilmoitetaan tiedoteikkunassa.

Lämpötilamuunnos, koodi

```
from Tkinter import *
import tkMessageBox

class Muunnin1:
    def __init__(self):
        self.paaikkuna = Tk()
        self.paaikkuna.title("Lampotilamuunnin")
        self.teksti = Label(self.paaikkuna, \
                            text = "Fahrenheit-lampotila:")
        self.ruutu = Entry(self.paaikkuna, \
                           width = 10)
        self.nappi = Button(self.paaikkuna, \
                            text = "Muunna", \
                            command = self.muunna_lampotila)
```

Lämpötilamuunnos, koodi jatkuu

```
        self.teksti.pack()
        self.ruutu.pack()
        self.nappi.pack()
        mainloop()

    def muunna_lampotila(self):
        syote = self.ruutu.get()
        fahrenheit = float(syote)
        celsius = 5.0 / 9.0 * (fahrenheit - 32.0)
        vastausteksti = "Lampotila on %.2f C" % (celsius)
        tkMessageBox.showinfo("Vastaus", vastausteksti)


oma_ikkuna = Muunnin1()
```

Virheenkäsittely lämpötilamuunnokseen

- Muutetaan lämpötilamuunnoksen tekevää ohjelmaa niin, että se käsittelee mahdolliset virheelliset syötteet.
- Virheestä ilmoitetaan virhedialogi-ikkunalla. Sellainen voidaan tehdä moduulin `tkMessageBox` funktion `showerror` avulla.

Virheenkäsittely lämpötilamuunnokseen, koodi

```
from Tkinter import *
import tkMessageBox

class Muunnin2:
    def __init__(self):
        self.paaikkuna = Tk()
        self.paaikkuna.title("Lampotilamuunnin")
        self.teksti = Label(self.paaikkuna, \
                            text = "Fahrenheit-lampotila:")
        self.ruutu = Entry(self.paaikkuna, \
                           width = 10)
        self.nappi = Button(self.paaikkuna, \
                            text = "Muunna", \
                            command = self.muunna_lampotila)
        self.teksti.pack()
        self.ruutu.pack()
```

Virheenkäsittely lämpötilamuunnokseen, koodi jatkuu

```
        self.nappi.pack()
        mainloop()

    def muunna_lampotila(self):
        syote = self.ruutu.get()
        try:
            fahrenheit = float(syote)
            celsius = 5.0 / 9.0 * (fahrenheit - 32.0)
            vastausteksti = "Lampotila on %.2f C" % (celsius)
            tkMessageBox.showinfo("Vastaus", vastausteksti)
        except ValueError:
            virheteksti = "Anna lampotila lukuna!"
            tkMessageBox.showerror("Virhe", virheteksti)


oma_ikkuna = Muunnin2()
```

Piirtäminen

- Luokan `Canvas` avulla voidaan luoda piirtoalusta, jolle voidaan piirtää erilaisia kuvioita, viivoja, tekstejä jne.
- Näin voidaan saada ohjelma tulostamaan erilaisia kaavioita tai kuvaajia.
- Piirtoalustan koordinaatteja lasketaan pikseleiden avulla. Vasemman yläkulman koordinaatit ovat `(0, 0)`
- Joitakin luokan `Canvas` metodeita. Mahdollisia parametreja on paljon erilaisia, mutta ensimmäiset parametrit kertovat ne koordinaatit, mihin komponentti piirretään:
  - `create_line` piirtää suoran viivan
  - `create_polygon` piirtää murtoviivan
  - `create_rectangle` piirtää suorakulmion
  - `create_oval` piirtää ympyrän tai ellipsin
  - `create_text` piirtää tekstin

Esimerkki piirtävästä ohjelmasta

```
from Tkinter import *

class Piirtoikkuna:
    def __init__(self):
        self.paaikkuna = Tk()
        self.paaikkuna.title("Piirtoesimerkki")
        self.piirtoalusta = Canvas(self.paaikkuna, \
                                   width = 200, \
                                   height = 300, \
                                   background = "lightblue")
        self.piirtoalusta.pack()
        self.piirtoalusta.create_line(0, 0, 100, 100, \
                                      fill = "red")
        self.piirtoalusta.create_rectangle(50, 50, 150, 150,\
                                           fill = "blue")
```

Esimerkki piirtävästä ohjelmasta jatkuu

```
        self.piirtoalusta.create_oval(100, 100, 200, 200, \
                                      fill = "yellow")
        self.piirtoalusta.create_line(70, 30, 95, 180, \
                                      fill = "green")
        self.piirtoalusta.create_text(100, 250, \
                                      text = "Hurraa!", \
                                      fill = "brown")

        mainloop()

oma_ikkuna = Piirtoikkuna()
```

Skandinaaviset aakkoset ja muut erikoismerkit

- Jos Python-ohjelmassa haluaa käyttää esimerkiksi skandinaavisia aakkosia tai euron merkkiä, on ohjelmatiedoston alkuun lisättävä kommentti, joka kertoo, miten nämä merkit on koodattu.
- Koodaus tarkoittaa sitä, että mikä luku vastaa mitäkin merkkiä ohjelmatiedostoa tallennettaessa.
- Koodaus vaihtelee sen mukaan, millä järjestelmällä ja editorilla ohjelma on kirjoitettu. Alla joitakin yleisiä koodaustapoja vastaavia alkukommentteja.

```
# -*- coding: utf-8 -*-
# -*- coding: cp850 -*-
# -*- coding: cp1252 -*-
# -*- coding: iso-latin-1 -*-
# -*- coding: iso-latin-15 -*-
```