Skolan för Datavetenskap och kommunikation

DD1311 Programmeringsteknik med PBL

Föreläsning 15

Idag:

- Granskning
- Felhantering
- GUI

På torsdag:

- Mer om GUI
- På grupptimmen: genomgång av granskningsprotokollet

Granskning

- Innan du redovisar ditt program ska det granskas.
- Den som granskar ska fylla i granskningsprotokollet och avgöra om programmet är användarvänligt, flexibelt, mm
- Granskaren ska vara med vid redovisningen.
- Alla måste granska ett program!

Centrerad utskrift

- En metod som skriver ut centrerat:

```
... och vinnaren är ...
*****************************************
                  AISHA
              med balladen
     WHAT FOR? (ONLY MR GOD KNOWS WHY)
*****************************************
```

Exempel (inte flexibelt)

```
def vinnarinfo(self):
   stjarnor = '*'*50
   print stjarnor
   print (self.artist.upper()).center(50)
   print "med balladen".center(50)
   print (self.titel.upper()).center(50)
   print stjarnor
```

Med parameter

```
def vinnarinfo(self, pos):
   stjarnor = '*'*pos
   print stjarnor
   print (self.artist.upper()).center(pos)
   print "med balladen".center(pos)
   print (self.titel.upper()).center(pos)
   print stjarnor
```

Mer om granskning på grupptimmen!

Algoritm för sökning

1. Skapa en tom lista (*funna*) för alla träffar.
2. Gå igenom varje element i listan
   - Om elementet är det sökta:
     Lägg till objektet till *funna*
3. Efteråt returnerar man *funna* med alla träffar.

Funktion för sökning

```
def sok(x,lista):
    """Söker efter låtskrivaren x i lista """
    funna = []
    for sang in lista:
        if (sang.text == x) or (sang.musik == x):
            funna.append(sang)
    return funna
```

```
ballader = sok("Bobby Ljunggren",schlagerfest)
alla = sok("Bobby Ljunggren",esc2010)
```

Felhantering, t ex

- Felaktig inmatning:
  - Tecken istället för tal
  - För stort/för litet tal
- Filer:
  - Infil saknas
  - Felaktiga data i filen
- Lista/dictionary:
  - Index saknas
  - Nyckel saknas

Exception - repetition

- När något blir fel i ett Python-program uppstår ett särfall, t ex NameError:

```
>>> print sko

Traceback (most recent call last):
  File "<pyshell#17>", line 1, in -toplevel-
    print sko
NameError: name 'sko' is not defined
```

- Man kan ta hand om särfall genom att införa `try-except-else`-satser för de delar i programmet som kan krascha.

Särfall - exempel

```
try:
    tal = input("Vad ska inverteras? ")
    invers = 1.0/tal
except (ZeroDivisionError):
    print "Noll kan inte inverteras"
except (NameError):
    print "Du borde ha skrivit ett tal!"
else:
    print "Inversen blev", invers
```

Exempel i slinga

```
def lasPengar():
   """ Läser in tills man ger ett heltal"""
   pengar = None
   while not pengar:
      try:
        svar = raw_input(”Ange belopp: ")
        pengar = int(svar)
      except(ValueError), e:
        print "Felaktigt belopp, försök igen"
   return pengar
```

Grafiskt användargränssnitt (GUI)

Använd modulen **Tkinter**, som har klasser för *komponenter*.

Se "Referenser" på kursens webbsida: GUI-länkar

Button	Menu
Canvas	Menubutton
Checkbutton	Message
Entry	Radiobutton
Frame	Scale
Label	Scrollbar
Listbox	Text

Hur gör man?

```
from Tkinter import *
roten = Tk()
knapp = Button(roten, text="Tryck")
knapp.pack()
roten.mainloop()
```

Hämtar alla klasser i modulen Tkinter.

Tk-konstruktorn - skapar rotfönstret.

Button-konstruktorn - skapar en knapp.

Knappen placeras.

Startar en slinga som väntar på inmatning från användaren.

Komponenter

- Knappar och annat kallas *komponenter* och är objekt.
- Varje komponent har en konstruktor med många defaultparametrar.
- Anropa bara med det som behövs:

```
knapp = Button(roten,text="Handla")
```

- Första parameter ska vara roten

Ändra attribut

- Attributen kan ändras ett i taget:

```
knapp["text"] = "Klart"
```

- Med metoden *config* kan man ändra flera attribut åt gången:

```
knapp.config(bg = "lightblue",
    height = 3, width = 9,
    font = ('times', 20, 'italic'))
```

- Här ändrar vi knappens färg, storlek, och textfont.

Variabler

- Ett attribut som alla komponenter har är `variable`.
- Om man i förväg skapat ett variabelobjekt:
  `s = StringVar()`
- så kan man koppla ihop variabel och komponent med
  `knapp["variable"] = s`
- Metoden `get` hämtar data från en variabel.

Anropa funktion med knapptryck!

- Ett annat attribut som alla komponenter har är `command`.
- Där anger man vilken metod/funktion som ska anropas när komponenten används.
- Om vi skriver en funktion addera() som ska anropas när nån trycker på knapp så kan vi koppla ihop funktion med knapp så här:
  `knapp["command"] = addera`

- Man kan också koppla ihop knappen med en funktion när den skapas:

```
from Tkinter import *

def byttext():
    knapp["text"] = "Aj!"

roten = Tk()
knapp = Button(roten,
               text = "Tryck inte",
               command = byttext)
knapp.pack()
roten.mainloop()
```

Layout

- Komponenter har metoder som styr hur de ska placeras i fönstret.
- Enklast är att använda pack:
  `knapp.pack()`
- Men bättre kontroll fås med grid:
  `knapp.grid(row=4, column=3)`
- Rita först en skiss över hur det ska se ut!

.py eller .pyw?

- Program som är avsedda att startas med dubbelklick på ikonen brukar man ge efternamnet ".pyw", t ex "saga.pyw"
- Då slipper man det svarta DOS-fönstret i bakgrunden.
- Men under avlusningen kan man utnyttja DOS-fönstret för kontrollutskrifter med print eller raw_input.

Tkinter och IDLE

- Din utvecklingsmiljö IDLE är skriven i Tkinter!
- Det innebär att *mainloop* redan är igång om du kör programmet inifrån IDLE, vilket kan få Shell-fönstret att bete sig underligt.
- Kommentera därför bort raden

  ```
  roten.mainloop()
  ```

  när du kör ditt Tkinter-program i IDLE.