Skolan för Datavetenskap och kommunikation

Programmeringsteknik

Föreläsning 15

Grafiska gränssnitt - GUI

Tkinter

Tillstånd

Komponenter

Layout

Händelser

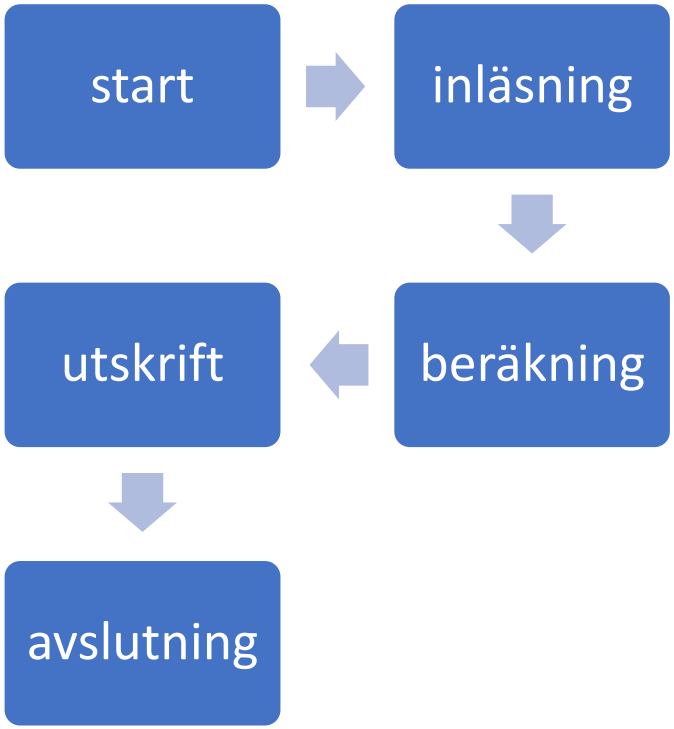
Länkar & exempel

- ...finns på Canvas, se "Exempel tkinter"

tkinter

- tkinter är ett av flera sätt att skriva grafiska gränssnitt i Python
- Vi använder tkinter därför att:
  - tkinter följer med Pythoninstallationen,
  - och introduceras i kursboken, se kap 13

textbaserat program



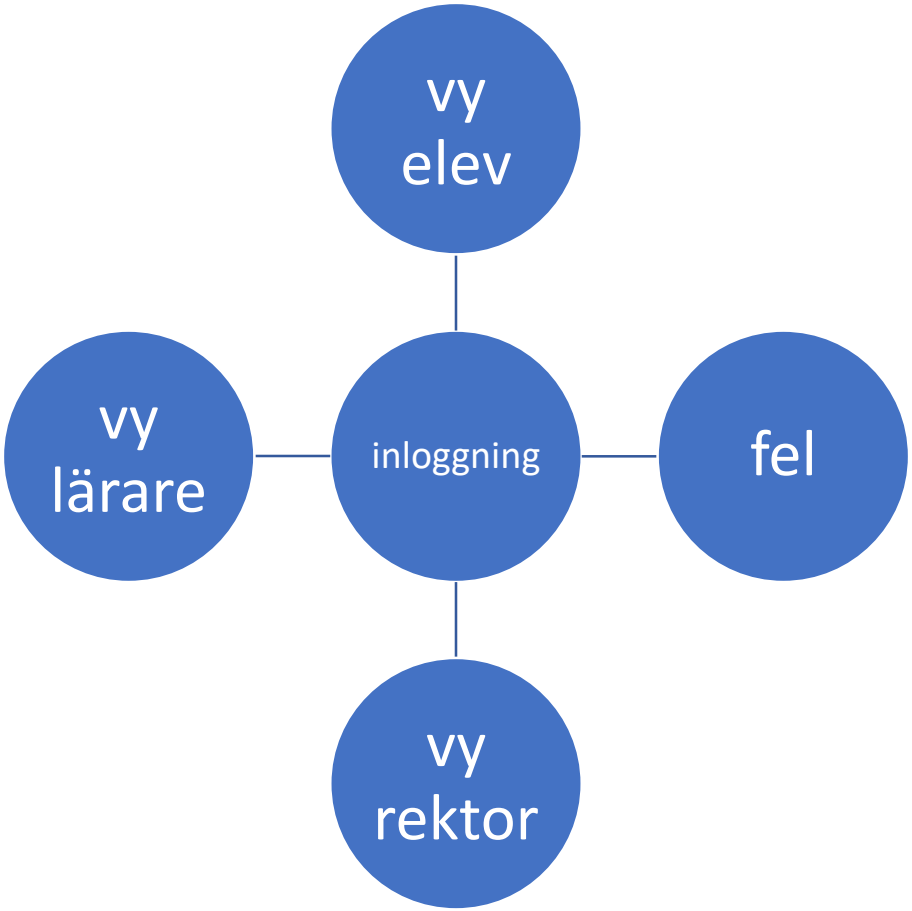
tillstånd

- Med ett grafiskt gränssnitt måste man tänka på vad som visas på skärmen i ett visst läge.
- Man brukar använda sig av *tillstånd* (state) när man modellerar hur programmet ska fungera.
- Programmets tillstånd beror av variablers värden, t ex

  användare = "elev"

- Vad ska synas för en elev? Hur skilkjer det sig från det läraren ser?

tillstånd - graf



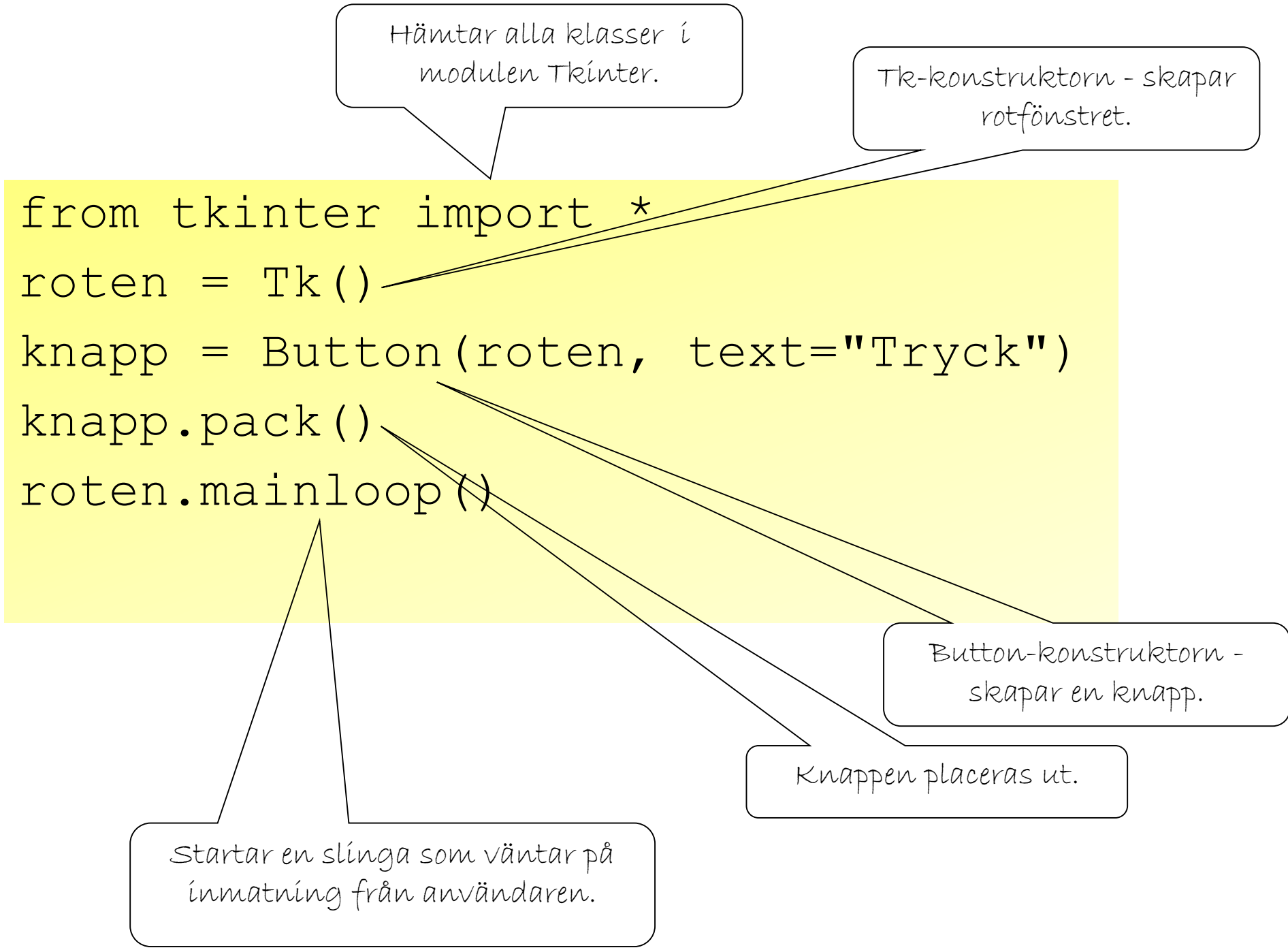
Komponenter

Modulen **tkinter**, har klasser för grafiska *komponenter*.

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

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

enklast.py


Hämtar alla klasser i modulen Tkinter.
Tk-konstruktorn - skapar rotfönstret.
from tkinter import *
roten = Tk()
knapp = Button(roten, text="Tryck")
knapp.pack()
roten.mainloop()
Button-konstruktorn - skapar en knapp.
Knappen placeras ut.
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

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

config

Med metoden *config* kan man ändra flera attribut

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

Här ändrar vi knappens färg, storlek, och font.

finare.py

command

- Ett 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
```

```
from tkinter import *

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

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

Parametrar då?

- Här skickar man bara med *namnet* på funktionen.
- Hur ska man göra om funktionen har parametrar?
- Lösning: Använd en lambda-funktion (ett sätt att skapa en funktion där den behövs)

Lambda-funktion

Vanlig funktion	Lambdafunktion
`def dubbla(x):` `  return 2*x`	`dubbla = lambda x : 2*x`

lambda i command

```
from tkinter import *

def byttext(t):
    knapp["text"] = t

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

byttext.py

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!

saga1.py ... saga7.py

	0	1	2	3
0	Rubrik			
1	Person:			
2	Sak:			
3	Verb:			
4	Adjektiv:			
5	Kroppsdel:			
6	Skriv saga			

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.

saga2.py

Felhantering

Olika orsaker till programkrasch:

Felaktig inmatning

- Tecken istället för tal
- För stort/för litet tal

Problem vid inläsning från fil

- Infil saknas
- Felaktiga data i filen

Problem med 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

```
print("Talinvertering")
tal = int(input("Ge ett tal: "))
try:
    invers = 1.0/tal
except ZeroDivisionError:
    print("Noll kan inte inverteras")
else:
    print("Inversen blev", invers)
```

Exempel i slinga

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

Felhantering i tkinter

- I messagebox finns "popupfönster" som lämpar sig för felhantering:
  - showinfo
  - showwarning
  - showerror
  - askquestion
  - askyesnocancel
- Alla tar två parametrar: title och message
- Vissa har returvärde (askyesnocancel)

Exempel

```
from tkinter import *
rot = Tk()
messagebox.showerror(title="Fel",   \
        message="Du har just gjort fel.")

rot.mainloop()
```

message.py