

*Skolan för Datavetenskap och kommunikation*

# PROGRAMMERINGSTEKNIK

---

FÖRELÄSNING 15

---

## Grafiska gränssnitt - GUI

Tkinter

Tillstånd

Komponenter

Layout

Händelser

# LÄNKAR & EXEMPEL

---

- ...finns på kurswebsidan under "GUI med 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

---



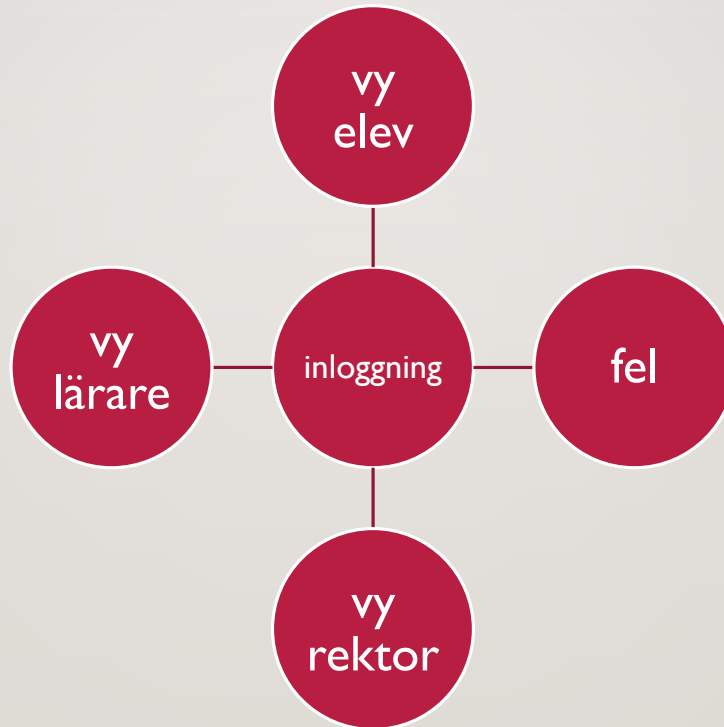
# TILLSTÅND

---

- 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 variabelers värden, t ex  
användare = "elev"

# 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()
```

# VAD GÖR VAD?

Hämtar alla klasser i  
modulen Tkinter.

```
from tkinter import *
```

Tk-konstruktorn - skapar  
rotfönstret.

```
roten = Tk()
```

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

Button-konstruktorn -  
skapar en knapp.

```
knapp.pack()
```

Knappen placeras ut.

```
roten.mainloop()
```

Startar en slinga som väntar på  
inmatning från användaren.

# KOMPONENTER

- Knappar och annat kallas *komponenter* och är objekt.
- Varje komponent har en konstruktör 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.

# 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 någon 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



# LAMBDA-FUNKTION...

- I Python kan man definiera en funktion med följande syntax:

*lambda parametrar: uttryck*

- Vanlig funktion:

```
def dubbla(x):  
    return 2*x
```

- Med lambda istället:

```
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()
```

# 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!
- Se programexemplet [saga.py](#)

0

1

2

3

Rubrík

Person:

Sak:

Verb:

Adjektív:

Kroppsdæl:

Skrív 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.

# 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
```

# 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: SHOWERROR

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