Skolan för Datavetenskap och kommunikation

Föreläsning 11

PROGRAMMERINGSTEKNIK

– läsa Pythons dokumentation
– referenser
– kopiera en lista
– klass-attribut
– rekursion

pythons dokumentation

- Hur vet man vilka funktioner, klasser, metoder och moduler som finns? Titta på sidan:

  http://docs.python.org/3/

Tutorial

- Som en kortfattad lärobok.
- Här finns många exempel, se t ex range

The given end point is never part of the generated sequence; range(10) generates 10 values, the legal indices for items of a sequence of length 10. It is possible to let the range start at another number, or to specify a different increment (even negative; sometimes this is called the 'step'):

```
range(5, 10)
    5 through 9

range(0, 10, 3)
    0, 3, 6, 9

range(-10, -100, -30)
    -10, -40, -70
```

The Python Standard Library

- Här finns alla detaljer, t ex vilka metoder strängar har.

4.7.1. String Methods¶

`str`**.endswith**(*suffix*[, *start*[, *end*]])

Return True if the string ends with the specified *suffix*, otherwise return False. *suffix* can also be a tuple of suffixes to look for. With optional *start*, test beginning at that position. With optional *end*, stop comparing at that position.

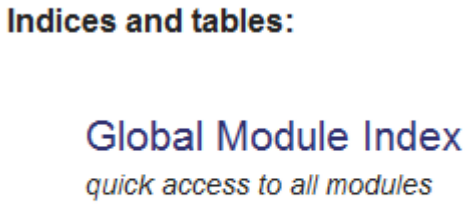
strategi

- Sök efter lämpliga metoder
- Kolla
  - Vilka parametrar har den?
  - Vilka är valfria parametrar? Står inom[]
  - Vilka returvärden har den?
  - Testa att anropa i shell-fönstret!

Global Module Index

- Hur vet man vilka moduler som finns?
- Se Global Module Index:

  random, copy, datetime, ...



help

Det finns också information direkt i Python (utan webbläsare)

```
import random
help(random)
help(random.choice)
```

modulen copy

Assignment statements in Python do not copy objects, they create bindings between a target and an object. For collections that are mutable or contain mutable items, a copy is sometimes needed so one can change one copy without changing the other.This module provides generic (shallow and deep) copying operations.

Interface summary:

```
copy.copy(x) Return a shallow copy of x.
copy.deepcopy(x) Return a deep copy of x.
```

tilldelning (repetition)

```
x = 5
y = x
y += 1
print("x=", x, "y=", y)
```

tilldelning lista, exempel

```
def main():
    svar = input("Vilka spel har du spelat? ")
    dina_spel = svar.strip().split()
    print("Jaså, du har spelat ", end ="")
    utskrift(dina_spel)
    mina_spel = dina_spel
    mina_spel.append("och Zork")
    print("Jag har spelat ", end ="")
    utskrift(mina_spel)
    print("Du hade visst bara spelat", end=" ")
    utskrift(dina_spel)
```

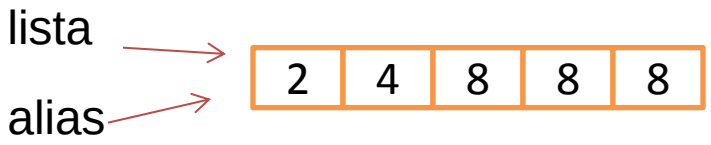
referenser

En listvariabel har en *referens* till listan.

Vid tilldelning är det referensen som kopieras - inte listelementen.

kopiera en lista

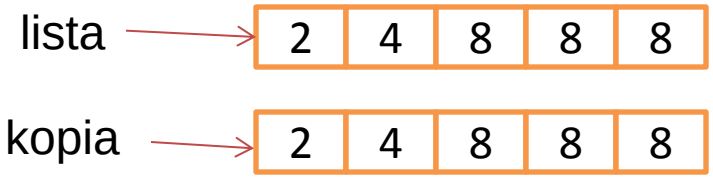
```
alias = lista
```



Vill man ha en *kopia* av hela listan kan man använda `copy` i modulen `copy`:

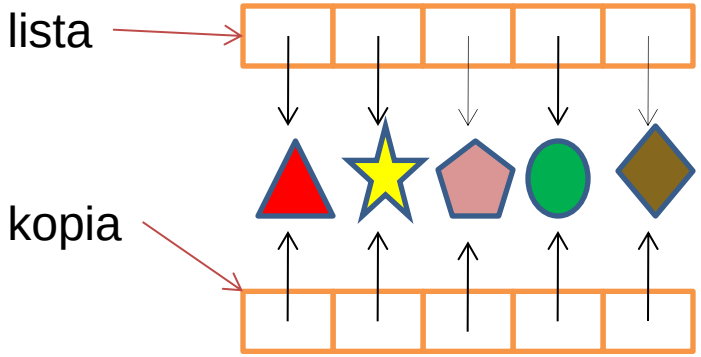
```
import copy

kopia = copy.copy(lista)
```



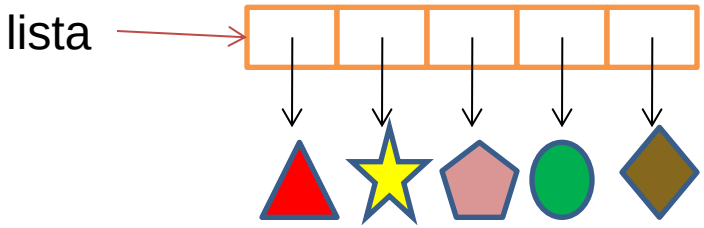
kopiera en lista av objekt

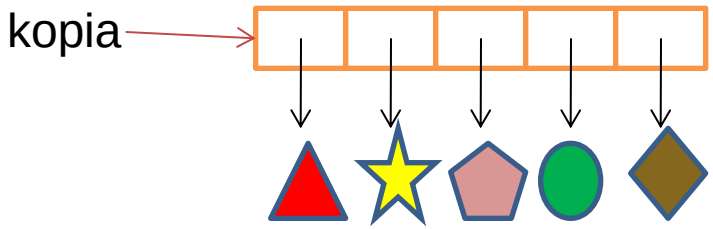
Om det är objekt i listan
kopierar copy.copy() *referenserna*
till varje objekt!



Använd *deepcopy* för att
även göra kopior av objekten.

```
import copy
kopia = copy.deepcopy(lista)
```





klass-attribut

Ett attribut som definieras i klassen, men utanför __init__, kallas för ett *klass-attribut.*

Varje objekt har sin egen uppsättning attribut, men det finns bara ett exemplar av klass-attributet.

```
class Tal:

   klassAttribut = 17

   def __init__(self):
      self.attribut = 42

objekt = Tal()
print(objekt.attribut)
print(Tal.klassAttribut)
```

rekursion

Rekursiva funktioner är enkla att skriva i Python.

Exempel: Fibonaccital

$F(0)=1$

$F(1)=1$

$F(n) = F(n-1)+F(n-2)$ för n>1

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```