Skolan för Datavetenskap och kommunikation

FÖRELÄSNING 10

PROGRAMMERINGSTEKNIK

- läsa Pythons dokumentation
- referenser
- kopiera en lista
- klass-attribut
- rekursion (intro)

PYTHONS DOKUMENTATION

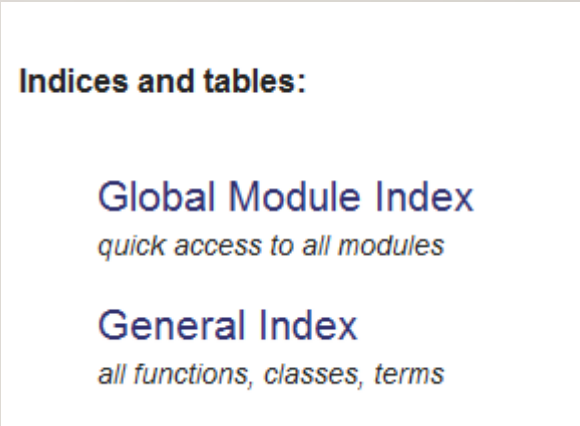
- Hur vet man vilka moduler som finns? Vilka metoder?
- Titta på sidan:

  http://docs.python.org/3/
- Under Global Module Index:

  moduler, t ex random, copy, datetime
- Under General Index:

  metoder för inbyggda klasser, t ex

  string, list



HELP

Det finns också information direkt i Python (utan webbläsare)

```
import random

help(random)

help(random.choice)
```

EXEMPEL: MODULEN **COPY**

Assignment statements in Python do not copy objects, they create bindings between a targ
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

```
x = 5

y = x

y += 1

print("x=", x, "y=", y)
```

TILLDELNING LISTA

```
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)
```

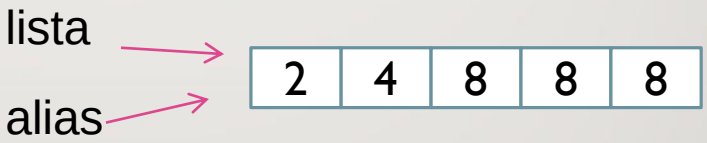
REFERENSER

En listvariabel har en *referens* till listan.

Vid tilldelning är det referensen som kopieras - inte listelementen.

KOPIERA EN LISTA

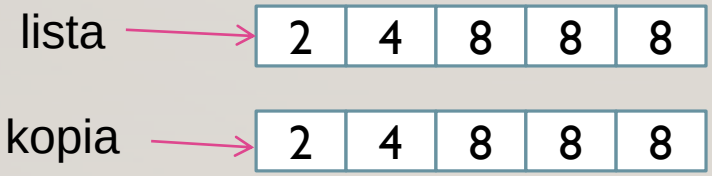
```
alias = lista
```



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

```
import copy

kopia = copy.copy(lista)
```

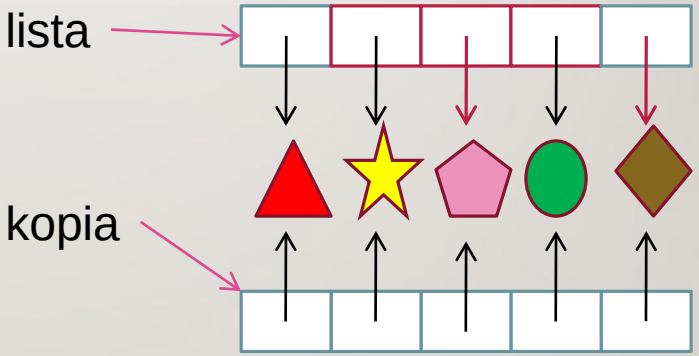


KOPIERA EN LISTA AV OBJEKT

Om det är objekt i listan

kopierar copy.copy() *referenserna*

till varje objekt!

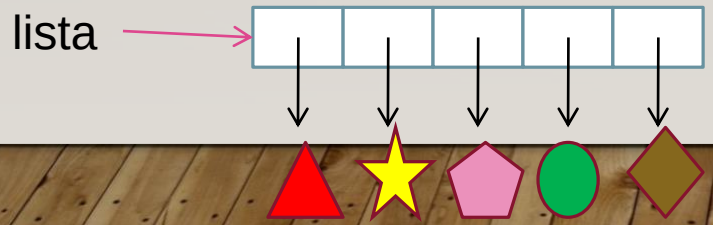


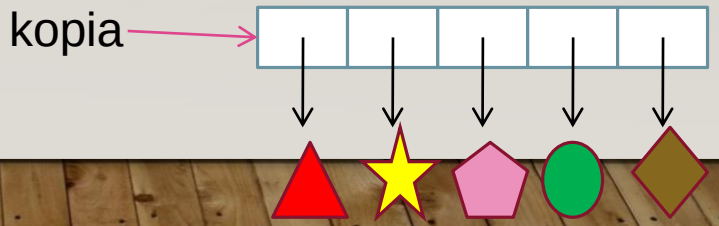
Använd *deepcopy* för att

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)
```