

GruProg DD1331

Övning 2

Anna Karlhede

karlhed@kth.se

Repo: <https://github.com/TypAnna/GruProgDD1331>

Idag

- Repetition
 - (o)föränderliga datatyper
 - key-words
 - if-satser
 - for- och while-loopar
 - något om metoder
 - listor (arrays i boken)
- Uppgifter

Att lära sig programmera

Att lära sig att programmera handlar mycket om att lära sig ett nytt sätt att tänka och lösa uppgifter på.

Några gyllene principer att förhålla sig till:

- Gör en övergripande plan för vad du behöver göra - rita gärna!
 - Observera att du inte behöver veta exakt *hur* du ska göra allt från början
- Bryt ner problemet i sina minsta beståndsdelar
 - Vad är det absolut första/simplaste du behöver göra?
- Skriv programmet så att det är lätt för andra att förstå!
 - Bra och tydliga variabelnamn
 - Kan du inte sammanfatta en metod i en mening så gör den mindre
- Om du sitter fast - beskriv vad programmet gör steg för steg, antingen för en kompis, eller för dig själv (högt!). Fantastiskt hur många problem/buggar som kan lösas på detta sätt.

Repetition



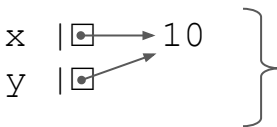
Datatyper - oföränderliga

Oföränderliga datatyper, `immutable`

- `int` - heltal
- `string` - sammansättning av tecken (en text), ex: "hejhej", "123", "123!!2#"
- `bool` - sanningsvärde, är antingen `true` (1) eller `false` (0)
- `float` - decimaltal
- `tuple` - som en lista men med fix längd och vars element du ej kan byta ut

Immutability exempel

```
x = 10
y = x
```



The diagram shows two variables, x and y, each with a small box containing a dot. Arrows from both boxes point to the number 10. A large curly brace on the right groups these two arrows together.

x och y pekar på samma objekt, samma plats i minnet!

```
x = x + 1
```



The diagram shows a single variable x with a small box containing a dot. An arrow from the box points to the number 11. A large curly brace on the right groups this arrow.

Ett nytt `int` objekt skapas med värdet 11 och tilldelas (binds till) variabeln x.
y är oförändrat och pekar mot samma plats i minnet som förut.

Datatyper - föränderliga

Föränderliga datatyper, `mutable`.

- listor ex: `[1, 5, 10]`, `["hej", "på", "dig"]`
 - kan bestå av blandade typer, ex: `[17, "katt"]`

När vi ändrar på muterbara objekt så skapas inget nytt objekt (som det gjordes på förra sliden).

```
aList = [1, 2 , 3]
bList = aList //aList och bList pekar mot samma objekt
aList[0] = 5 //både aList OCH bList är nu [5, 2, 3]!!
```

Tänk på detta medan ni programmerar - kan ge upphov till många svårförklarliga buggar ;-;

Keywords

I alla språk finns det s.k. “keywords” som har en viss (reserverad) funktion. I python har vi ex:

- and
- if
- for
- while
- return
- break
- def
- True
- och [många fler](#)...

Dessa kan inte användas som variabel- eller funktions-namn utan har en viss (specifik) betydelse i språket.

Conditionals

```
if len(someList) == 4:
    print(`The list has 4 elements`)
elif len(someList) == 5:
    print(`The list has 5 elements`)
elif len(someList) < 4:
    print(`The list has less than 4 elements`)
else:
    print(`The list has more than 5 elements`)
```

} En if-sats kan ha hur många elif (else if) klausuler som helst...

} ...men bara en else-klausul, som fångar alla "övriga fall"

Programmet kommer bara printa *en* av dessa rader, och sen hoppa till slutet av if-satsen och fortsätta med den kod som står där.

while-loop

Så länge uttrycket specificerat efter while gäller, kör koden som står i det efterföljande blocket.
Obs: ett block specificeras av indentering!!!

```
i = 1
total = 0
while i < 10:
    total = total + i
    i += 1 //vad händer om vi inte har med denna rad?
print(`The sum of the first 10 integers is: `, total)
```

Kan avbryta en while-sats genom att använda nyckelordet **break**. Detta görs förslagsvis om ett visst (något) villkor är uppfyllt.

Vi kan också använda nyckelordet **continue** om vi vill avbryta den nuvarande iterationen av while-loopen och fortsätta med nästa.

for-loop

En for-loop itererar över en mängd. Detta kan vara en mängd heltal, från 0 till 9 likt den förra sliden. Metoden blir då något mer kompakt.

```
total = 0
for i in range(10):
    total += i
print(`The sum of the first 10 integers (starting with 0) is :`, total)
```

Men en for-loop kan också iterera över en array (och alla andra itererbara objekt, tex strängar..)! Om vi har en lista med heltal, och vill ta reda på vilka av dem som är delbara med 3 kan vi skriva följande:

```
aList = [183754, 8876401, 908172, 18]
for element in aList:
    if element % 3 == 0:
        print(str(element) + ` can be divided by 3`)
```

while vs for

- En while-loop itererar så länge något condition inte är false.
- En for-loop itererar istället över en samling objekt, och utför något för vart och ett av dessa element.

Något om metoder

Vi definierar en metod med keywordet `def`. Vi specificerar vad som hör till metoden med indentering! Det är stor skillnad på att definiera en metod, och anropa den.

```
def sayHi(name): #här definierar vi metoden sayHi som tar parametern name
    print("Hi", name) #detta är del av metoden
```

```
print("Hej alla") #detta är INTE en del av metoden!
```

```
sayHi("Arvid") #här anropar vi metoden - nu körs den!
```

Så fort vi har någonting som efterföljs av en parentes så är det en metod som körs. Det som står inuti är metodens parametrar. Ex:

- `input("Skriv ett tal")`
- `print("Hello world")`
- `calculateAverageAge(age1, age2)` #en metod kan ta flera parametrar

PS - Om du inte kan beskriva vad din funktion gör i en mening - gör den mindre!

Listor - en typ av datastruktur

Listor kommer med massa [inbyggda funktioner](#) i python, ex:

- `aList.append(x)`
- `len(aList)`
- `aList.reverse()` #obs - ändrar på listan

Kan ha listor i listor - `nameAndAge = [["Mei-Li", 24], ["Anne", 65], ...]`

En matris kan representeras på detta sätt.

```
aMatrix = [[1, 2, 3], [4, 5, 6]]  
aMatrix[0] ger den första raden
```

1	2	3
4	5	6

Uppgifter



Uppgift 1 - Beräkna arean av en triangel

Skriv ett program som frågar efter a , b och c samt beräknar och skriver ut triangelytan. Låt programmet även göra kontroller av a , b , c så att de angivna värdena verkligen kan bilda en triangel.

Uppgift 2 - Beräkna siffersumman

Skriv ett program som beräknar siffersumman av ett tresiffrigt tal.

Börja **inte** med att fundera på den exakta syntaxen.

Istället - fundera övergripande över vad som behövs göras...

Huvuddelen av uppgiften är att beräkna siffersumman - hur gör vi det rent matematiskt?

Vad kan gå fel?...

Kan tyckas onödigt att göra på detta sätt på en sån här "liten" uppgift - men det är tänket som är viktigt! Blir allt viktigare ju större uppgifter ni gör.

Uppgift 3 - Beräkna siffersumman

Skriv ett program som beräknar siffersumman av ett **godtyckligt** stort tal.

- while-loop?
- for-loop? - Kom ihåg att strängar är itererbara objekt!

Finns flera möjliga approacher...

Uppgift 4 - Hitta duplicates i en lista

Givet en lista, se om det finns några element som upprepas och returnera dessa element i en ny lista

En approach skulle kunna vara att titta på det första elementet i listan - och sedan titta igenom resten av listan och se om något av resterande element är lika med det första. Denna process kan vi upprepa för vart och ett av elementen.

MEN - är mycket kostsamt...

Uppgift 5 - Rita ut ett schackbräde

Skriv en metod som tar en parameter på storleken på ett schackbräde och skriv ut det i terminalen, där "*" representerar en svart ruta, och " " en vit ruta.