

# Introduktion till programmering

Funktioner

# Dagens föreläsning

1. Programmering – snabb repetition
2. Funktioner inom programmering
3. Moduler i Python och programmering

**Frågor innan vi börjar?**

# Programmering

- Programmering går ut på att ge instruktioner till dator
- Datorn gör det du säger till den, och inget annat
- Datorn behöver specifika instruktioner



# Exempel på instruktioner

Matematiska instruktioner (beräkningar)

```
>>> 5 + 5
```

```
10
```

```
>>> 10 * 2
```

```
20
```

```
>>> 20 / 4
```

```
5
```

# Exempel på instruktioner

Instruktioner att skriva ut saker (output)

```
>>> print("Hello World")  
Hello World  
>>> print("Tjena kexet, sitter du här och smular?")  
Tjena kexet, sitter du här och smular?
```

# Exempel på instruktioner

Instruktion att hämta data från användare (input)

```
>>> input("Vad heter du?")
Vad heter du?Anton
'Anton'
>>> input("Vilket är Sveriges bästa fotbollslag?")
Vilket är Sveriges bästa fotbollslag?Elfsborg
'Elfsborg'
```

```

0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e

```

```

# Asks user for their name
name = input("Please enter your name :")
if len(name) < 4:
    # If the users name has less than 4 characters
    print(name + " is a short name")
elif len(name) < 7:
    # If the users name has less than 7 characters (and more than 3)
    print(name + " is a medium long name")
else:
    # If the users name has mote than 6 characters
    print(name + " is a long name")

```



En **lat** programmerare är  
en **bra** programmerare



# Funktioner – att återanvända kod

- En stor del i programmering går ut på att vi vill göra:
  - Återanvändbar kod
  - Kod som är enkelt att felsöka
  - Kod som är enkel att underhålla
- Detta för att:
  - Spara tid
  - Spara pengar
  - Spara energi



```
# Beräkna priset i Dollar
kronor_per_dollar = 10.97
pris = round(1690 / kronor_per_dollar, 2)
print(pris)
```

VIC  
Logga in / Skapa nytt  
MITT KONTO ▾

ör ▾ Favoriter

```
# Beräkna priset i Dollar
kronor_per_dollar = 10.97
pris = round(990 / kronor_per_dollar, 2)
print(pris)
```

Kundvagnen är tom ▾  
Inkl. moms



**-15%**  
**1 690:-**  
Ord. pris 1 990 kr  
★★★★★

### Apple TV 4K 32GB

Med Apple TV 4K kan du titta på film och tv-program i fantastisk 4K HDR-kvalitet.

78 kvar

Lägg i k



**-14%**  
**5 990:-**  
Ord. pris 6 990 kr  
★★★★★

### HP 250 G6 + Rygsäck Core i5 8GB 256GB 15.6

Kraftfull bärbar med högupplösningsskärm och stor SSD-hårddisk. Just nu medföljer exklusiv rygsäck från HP.

```
# Beräkna priset i Dollar
kronor_per_dollar = 10.97
pris = round(5990 / kronor_per_dollar, 2)
print(pris)
```



**-23%**  
**999:-**  
Ord. pris 1 299 kr

### HP LaserJet Pro M130nw

- Tillverkare HP Inc.
- Typ Skrivare/kopiator/skanner
- Utskriftsteknik Laser
- Utskriftstyp Monokrom

72 kvar

Lägg i k



**-18%**  
**6 990:-**  
Ord. pris 8 490 kr

### HP 250 G6 Core i7 8GB 256GB 15.6

Hållbar mobil design

- Processortyp Core i7
- Processorgeneration 7
- Installerad minnesstorlek 8 GB
- Lagringskapacitet 256 GB

```
# Beräkna priset i Dollar
kronor_per_dollar = 10.97
pris = round(6990 / kronor_per_dollar, 2)
print(pris)
```



**-14%**  
**2 990:-**  
Ord. pris 3 490 kr  
★★★★★

### Apple iPhone SE 32GB Rymdgrå

Ett stort steg för den lilla.

- Skärmstorlek 4 tum
- Internminneskapacitet 32 GB
- Bildskärmsupplösning 1136 x 640 pixlar



**-13%**  
**3 990:-**  
Ord. pris 4 590 kr  
★★★★★

### Lenovo V110 Core i3 8GB 128GB 15.6

Snabb och prisvärd arbetsdator som passar din budget.



**-27%**  
**2 199:-**  
Ord. pris 2 999 kr

### Netgear Arlo VMS3230 - Base Station & 2 Cameras Vit

Trådlös övervakning för ditt hem/kontor

- Tillverkare NETGEAR
- Anslutningsteknik Trådlös



**-40%**  
**449:-**  
Ord. pris 745 kr

### Symantec Norton Security Deluxe 3.0 Nordic - Vid Köp Av Dator, Tablet Eller Mobil

Kampanjartikel - Får köpas vid samtidigt köp av dator/smartphone och extern hårddisk

*Functions should do one thing.  
They should do it well.  
They should do it only.*

# Inbyggda funktioner i Python

- Det finns många inbyggda funktioner i Python, vissa har vi redan använt. De är uppbyggda på samma sätt:
  - `input()` Hämtar en sträng från användaren
  - `type()` Berättar vilken datatyp ett värde är
  - `int()` Gör om ett värde till ett nummer
  - `str()` Gör om ett värde till sträng
- Man anger alltså namnet på den uppsättning instruktioner som vi vill använda

Vi vet vad de inbyggda  
funktionerna  
**gör** men inte **hur**

```
1 # Kontrollera vilken datatyp ett värde är
2 print(type(2))
3 print(type("Hej"))
4 print(type(True))
5 print(type(27.5))
6
```

PROBLEMS

TERMINAL

...

Python

✓

```
PS C:\Users\tsanti> & C:/Users/tsanti/AppData/Local/WindowsApps/python3.10.exe "c:/Users/tsanti/Box/Höreläsningar/2. Funktioner/demo.py"
```

```
<class 'int'>
```

```
<class 'str'>
```

```
<class 'bool'>
```

```
<class 'float'>
```

```
PS C:\Users\tsanti> 
```



```
1  """Byta datatyper för värden"""
2
3  value_100 = "100"
4  print("Variabeln: value_100 har datatypen:", type(value_100))
5
6  # Byter datatyp från sträng till heltal
7  value_100 = int(value_100)
8  print("Variabeln: value_100 har datatypen:", type(value_100))
9
```

PROBLEMS

OUTPUT

TERMINAL

PORTS

DEBUG CONSOLE

```
PS C:\Users\tsanti> & C:/Users/tsanti/AppData/Local/Microsoft/WindowsAp  
ar/2. Funktioner/demo.py"
```

```
Variabeln: value_100 har datatypen: <class 'str'>
```

```
Variabeln: value_100 har datatypen: <class 'int'>
```

```
PS C:\Users\tsanti> □
```

```
1  """Byta datatyper för värden"""
2
3  value_50 = 50
4  print("Variabeln: value_50 har datatypen:", type(value_50))
5
6  # Byter datatyp från heltal till sträng
7  value_50 = str(value_50)
8  print("Variabeln: value_50 har datatypen:", type(value_50))
9
```

PROBLEMS    OUTPUT    **TERMINAL**    PORTS    DEBUG CONSOLE

OUTPUT    TERMINAL    PORTS    DEBUG CONSOLE

TERMINAL    PORTS    DEBUG CONSOLE

PORTS      DEBUG CONSOLE

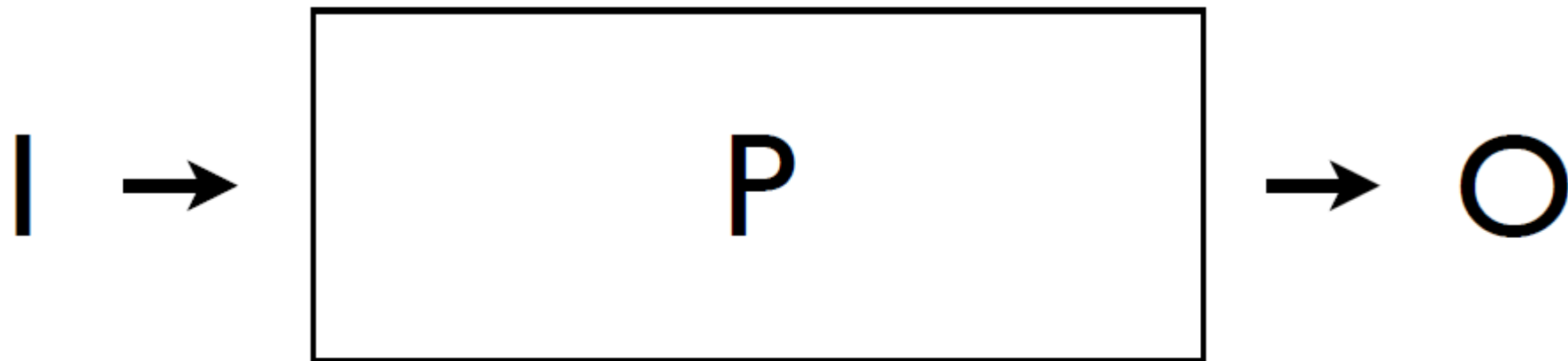
## DEBUG CONSOLE

```
PS C:\Users\tsanti> & C:/Users/tsanti/AppData/Local/Microsoft/WindowsApps/Powercat.exe -u 10.10.10.10 -p 4444 -s ar/2. Funktioner/demo.py"
```

```
Variabeln: value 50 har datatypen: <class 'int'>
```

```
Variabeln: value_50 har datatypen: <class 'str'>
```

```
PS C:\Users\tsanti>
```

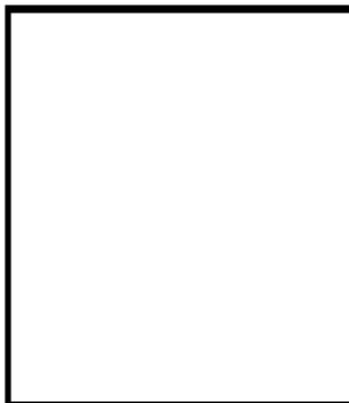


32

str

"32"

I



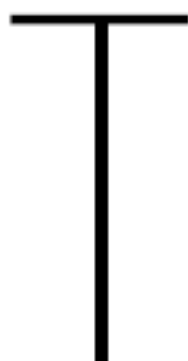
32



str(32)



funktionens namn



indata

= argument

## 2. Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

Built-in Functions				
<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

<https://docs.python.org/3/library/functions.html>

# En funktion löser ett problem

Är problemet stort?

**Gör flera mindre funktioner som löser delproblem!**

# Olika "typer" av funktioner

## 1. De som gör samma sak varje gång

- Detta är de absolut enklaste – de har vi inte tittat på än, men kommer att bygga snart.

## 2. De som gör olika saker beroende på hur vi kör dem

- `input("Här skickar vi med vad det ska stå när användaren ska skriva in en siffra: ")`
- `str(siffra)`
- `type(värde)`

## 3. De som returnerar saker (ett resultat)

- Funktionen "input", "type" och "str" returnerar ett resultat till oss.



**Att använda inbyggda  
funktioner**

**Att skapa egna funktioner**

The first rule of functions is that they  
should be small.

The second rule of functions is that  
*they should be smaller than that.*

# Att namnge instruktioner

- En funktion är en flera instruktioner som man namnger. Detta framförallt för att:
  - Kunna återanvända dessa instruktioner flera gånger
  - Kunna strukturera upp sin kod bättre
    - - Genom t.ex. olika filer
- Vi kan sedan köra dessa instruktioner genom att ange deras namn.

# Att skapa en funktion, och köra den

```
===== RESTART: C:/Users/TSANTII/Desktop/le2.py =====  
-----  
Welcome  
-----
```

```
print ("_"*40)  
print ("Welcome")  
print ("_"*40)
```



```
def welcome():  
    print ("_"*40)  
    print ("Welcome")  
    print ("_"*40)  
  
welcome()
```

# Skapa funktioner i olika språk

## Python

```
def welcome():  
    print("-"*40)  
    print("Welcome")  
    print("-"*40)
```

```
welcome()
```

## JavaScript

```
function welcome() {  
    document.writeln("-----");  
    document.writeln("Welcome");  
    document.writeln("-----");  
}  
  
welcome();
```

# Indentering?

- Till skillnad från många andra programmeringsspråk använder man inte "{" och "}" för att visa vad som tillhör en funktion, se t.ex. JavaScript:

```
function welcome() {  
    document.writeln("-----");  
    document.writeln("Welcome");  
    document.writeln("-----");  
}  
  
welcome();
```

- Istället så används ":" och indentering för att visa vad som tillhör funktionen:

```
def welcome():  
    print("-"*40)  
    print("Welcome")  
    print("-"*40)  
  
welcome()
```

*Any fool can write code that a computer can understand. Good programmers write code that humans can understand.*

– [Martin Fowler](#) (author and speaker on software development)



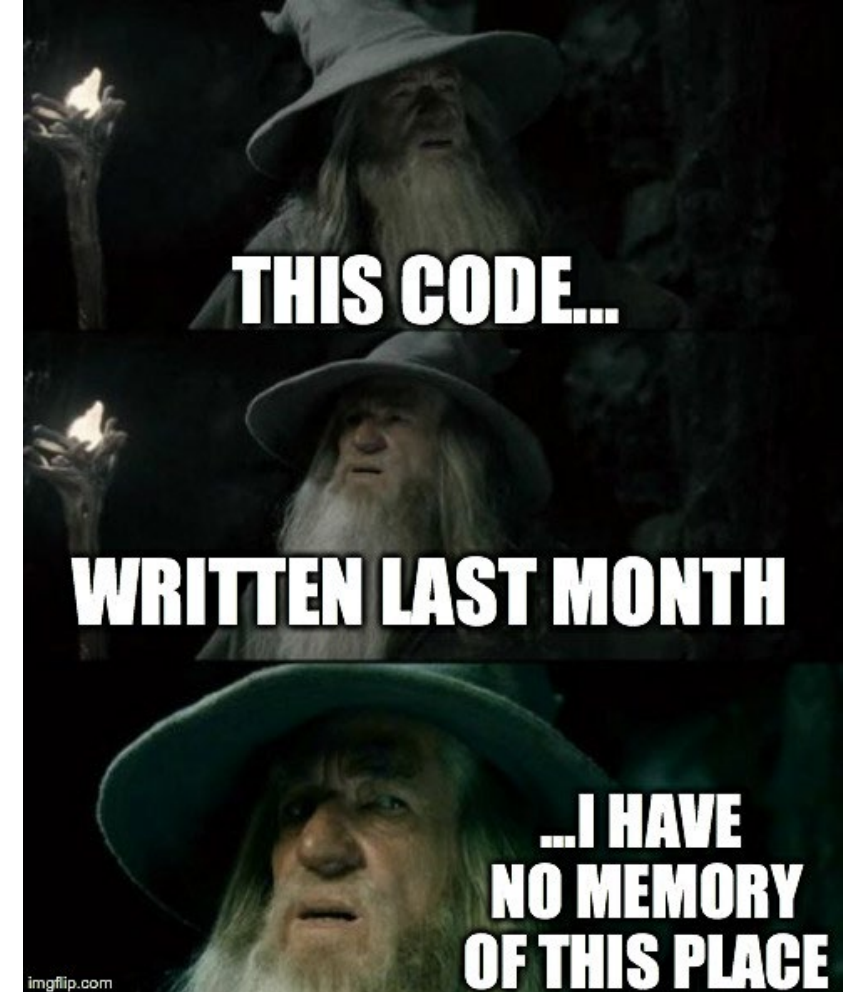
# Dokumentation för funktioner

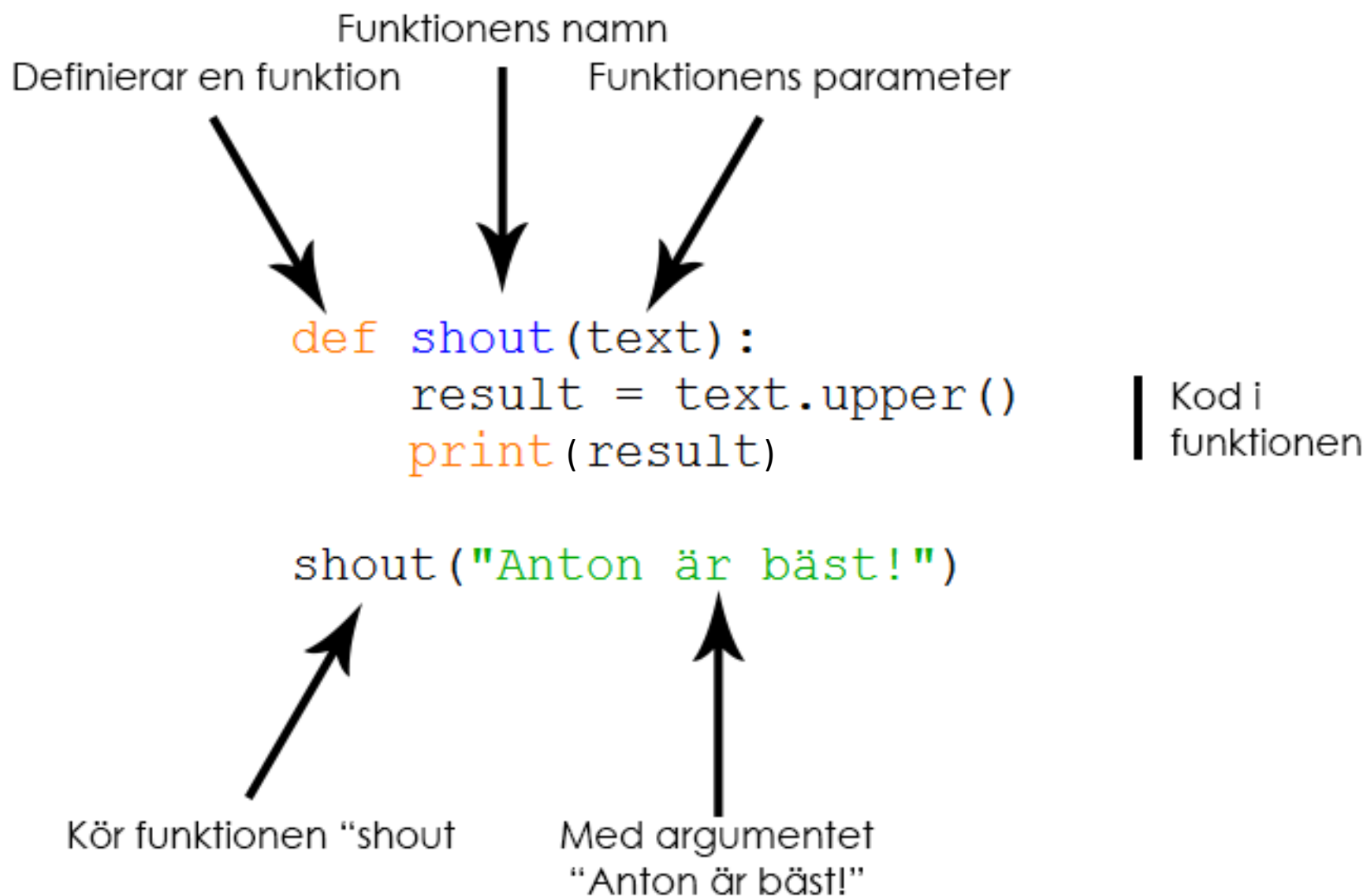
- Dokumentera vad funktionen gör
  - Eventuellt parametrar
  - Eventuellt returvärde (vi kommer till detta...)
- Detta görs genom ''' inom funktionen '''
- Vi kan ta reda på hur funktioner fungerar genom `help(funktionsnamn)`

```
def welcome():  
    '''  
    Welcomes the user to our program  
    '''  
    print("-"*40)  
    print("Welcome")  
    print("-"*40)
```

```
welcome()
```

```
>>> help(welcome)  
Help on function welcome in module __main__:  
  
welcome()  
    Welcomes the user to our program
```





# Case!

Bygga en lunchpresentatör

=====

Welcome!

=====

Lunch Monday is: Spagetti Bolognaise  
Lunch Tuesday is: Fish and chips  
Lunch Wednesday is: Hamburger  
Lunch Thursday is: Soup  
Lunch Friday is: Pizza

=====

Goodbye!

=====

```
print("="*40)
print("Welcome!")
print("="*40)
print("Lunch Monday is: Spagetti Bolognaise")
print("Lunch Tuesday is: Fish and chips")
print("Lunch Wednesday is: Hamburger")
print("Lunch Thursday is: Soup")
print("Lunch Friday is: Pizza")
print("="*40)
print("Goodbye!")
print("="*40)
```

```
=====
Welcome!
```

```
=====
Lunch Monday is: Spagetti Bolognaise
Lunch Tuesday is: Fish and chips
Lunch Wednesday is: Hamburger
Lunch Thursday is: Soup
Lunch Friday is: Pizza
```

```
=====
Goodbye!
```

1



```
=====
Welcome!
=====
```

2



```
Lunch Monday is: Spagetti Bolognaise
Lunch Tuesday is: Fish and chips
Lunch Wednesday is: Hamburger
Lunch Thursday is: Soup
Lunch Friday is: Pizza
```

3



```
=====
Goodbye!
=====
```

# Demo!

Bygga en lunchpresentatör

# Vad var bra med detta?

- Vi separerar vår kod till mindre bitar (funktioner)
  - Detta underlättar underhåll
  - Detta underlättar felsökning
- Varje funktion har **en** uppgift
  - welcome: Välkomna användaren
  - goodbye: Säga hejdå till användaren
  - weeks\_lunch: Presentera dagens lunch
- Vi kör våra funktioner sist i filen
  - Vi har programmets struktur väldigt tydligt – utan att egentligen veta vad varje funktion exakt gör



Vad kan det finnas för  
**problem** med detta?

# Vanligaste problemen med funktioner

- Tänk på att koden läses *uppfifrån och ner*. Detta innebär att vi kan inte kalla på en funktion innan den är deklarerad (skriven). Då får man detta fel:

```
welcome()
```

```
def welcome():
```

```
    '''
```

```
    Welcomes the user to our program
```

```
    '''
```

```
    print("-"*40)
```

```
    print("Welcome")
```

```
    print("-"*40)
```

```
Traceback (most recent call last):
```

```
  File "C:\Users\Anton\Box Sync\HT 2015\DA354A - Introduktion till programmering  
  \Föreläsningar\Föreläsning 2 - Funktioner\Exempel\1-Funktioner.py", line 1, in <  
  module>
```

```
    welcome()
```

```
NameError: name 'welcome' is not defined
```

- Vi glömmer att indentera koden i vår funktion. Då får man detta fel:

```
def welcome():
```

```
    '''
```

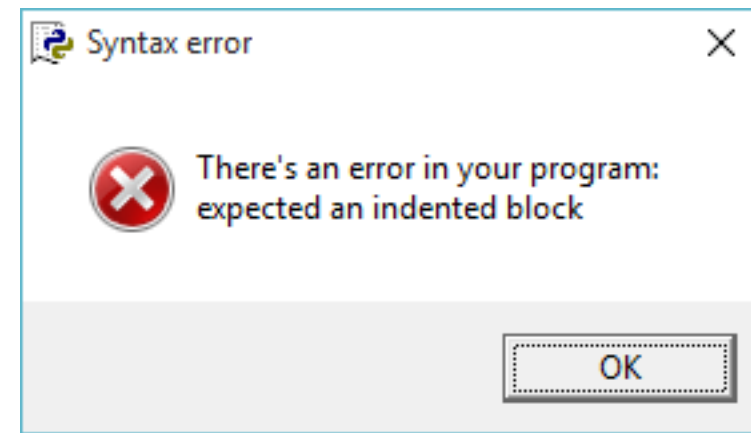
```
    Welcomes the user to our program
```

```
    '''
```

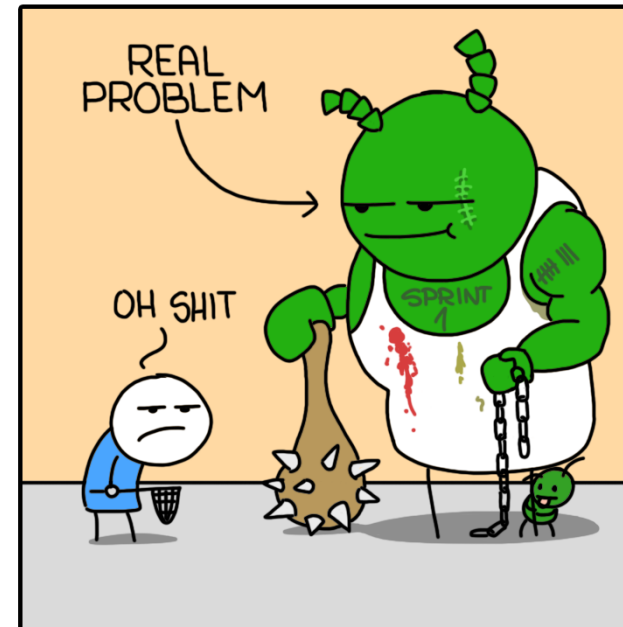
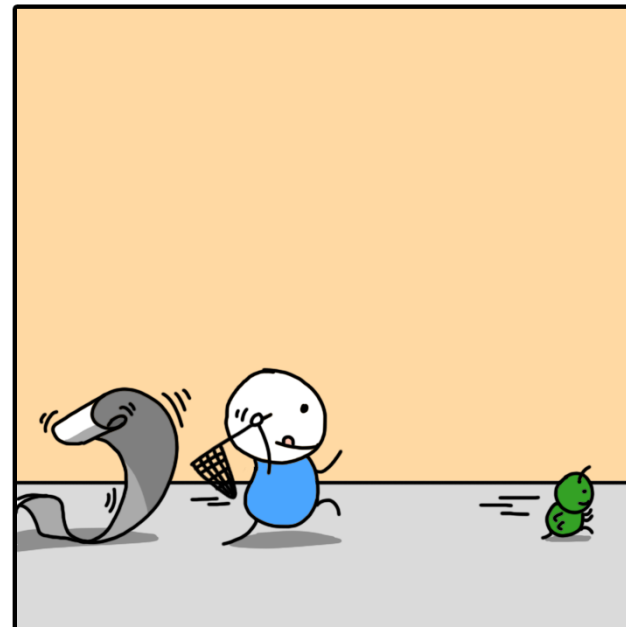
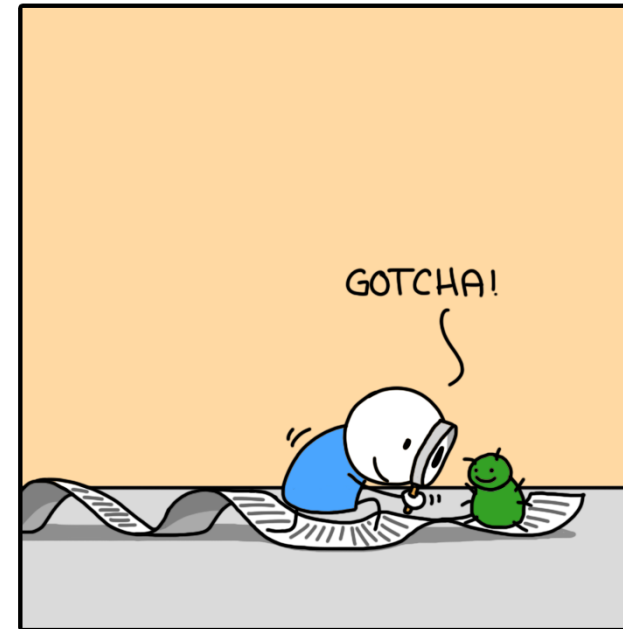
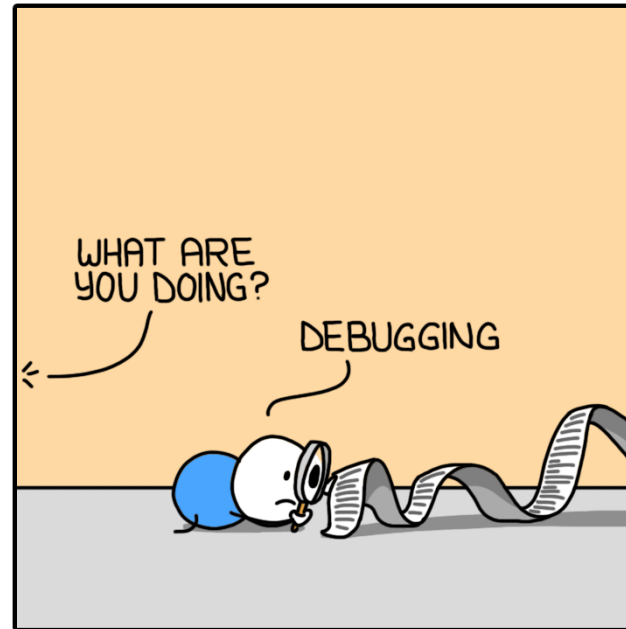
```
    print("-"*40)
```

```
    print("Welcome")
```

```
    print("-"*40)
```



# ROOT CAUSE



# Spåra fel till funktioner



```
def welcome():
    print("="*40)
    print("Welcome!")
    print("="*40)

def goodbye():
    print("="*40)
    print("Goodbye!")
    print("="*39+1)

def weeks_lunch():
    print("Lunch Monday is: Spagetti Bolognaise")
    print("Lunch Tuesday is: Fish and chips")
    print("Lunch Wednesday is: Hamburger")
    print("Lunch Thursday is: Soup")
    print("Lunch Friday is: Pizza")

# Run the program
welcome()
weeks_lunch()
goodbye()
```

```
Traceback (most recent call last):
  File "C:\Users\Anton\Box Sync\HT 2016\DA354A\Föreläsningar\Föreläsning 2 - Funktioner\Exempel\1-Funktioner.py", line 33, in <module>
    goodbye()
  File "C:\Users\Anton\Box Sync\HT 2016\DA354A\Föreläsningar\Föreläsning 2 - Funktioner\Exempel\1-Funktioner.py", line 22, in goodbye
    print("="*40+4)
TypeError: Can't convert 'int' object to str implicitly
>>>
```

Hey, are you sleeping?



Yes, now shut up



I know how to fix that bug on line 255



# Kunna effektivisera vår kod?

```
def welcome():  
    print("="*40)  
    print("Welcome!")  
    print("="*40)
```

```
def goodbye():  
    print("="*40)  
    print("Goodbye!")  
    print("="*40)
```

```
def weeks_lunch():  
    print("Lunch Monday is: Spagetti Bolognaise")  
    print("Lunch Tuesday is: Fish and chips")  
    print("Lunch Wednesday is: Hamburger")  
    print("Lunch Thursday is: Soup")  
    print("Lunch Friday is: Pizza")
```



Dessa liknar ju  
varandra väldigt  
mycket!

# Generella funktioner

- Vi vill ju skriva funktioner som kan återanvändas i så stor utsträckning som möjligt
- Detta göra man ofta genom att skicka med data (värden) till funktionen man vill köra.
  - Detta har vi redan testat – utan att vi tänkt på det, t.ex. genom:  
`input("meddelande")`
  - `type(värde)`  
`str(nummer)`  
etc.

```
nr_1 = input("Ange en siffra: ")  
nr_2 = input("Ange en siffra igen: ")  
nr_3 = input("Ange ytterliggare en siffra: ")
```



```
Ange en siffra: 10  
Ange en siffra igen: 20  
Ange ytterliggare en siffra: 30
```

```
>>> type("Hej")  
<class 'str'>  
>>> type(5)  
<class 'int'>  
>>> type(False)  
<class 'bool'>
```



# Generell rubrik-funktion

```
def welcome():  
    print("="*40)  
    print("Welcome!")  
    print("="*40)
```

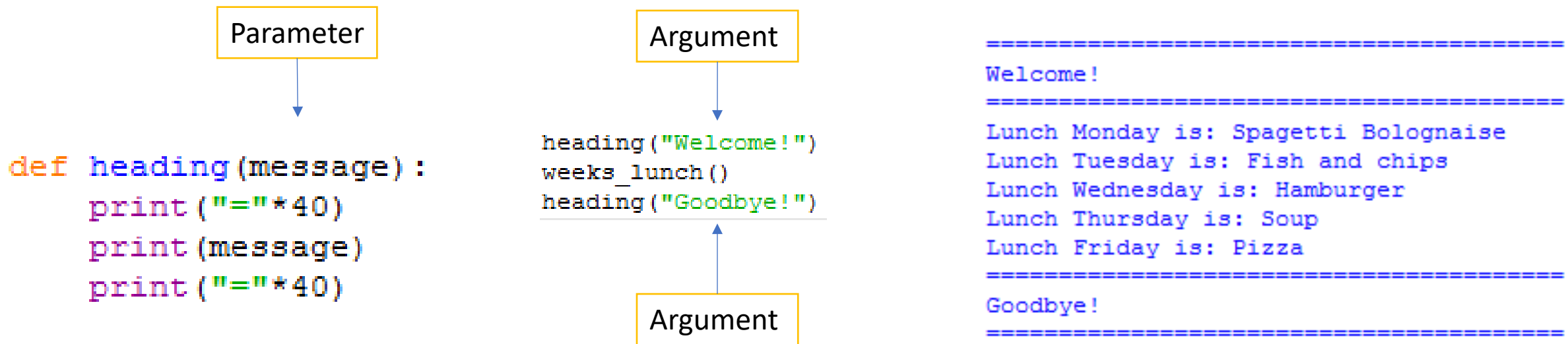
```
def goodbye():  
    print("="*40)  
    print("Goodbye!")  
    print("="*40)
```



```
def heading(message):  
    print("="*40)  
    print(message)  
    print("="*40)
```

# Argument och parametrar

- I exemplet tidigare skapade vi en funktion som hade **parametrar**. Detta innebär att vi kan – beroende på vad vi skickar med till funktionen när vi kör den – påverka resultatet.
- **Argument** kallas det värde som vi skickar med till funktionen.

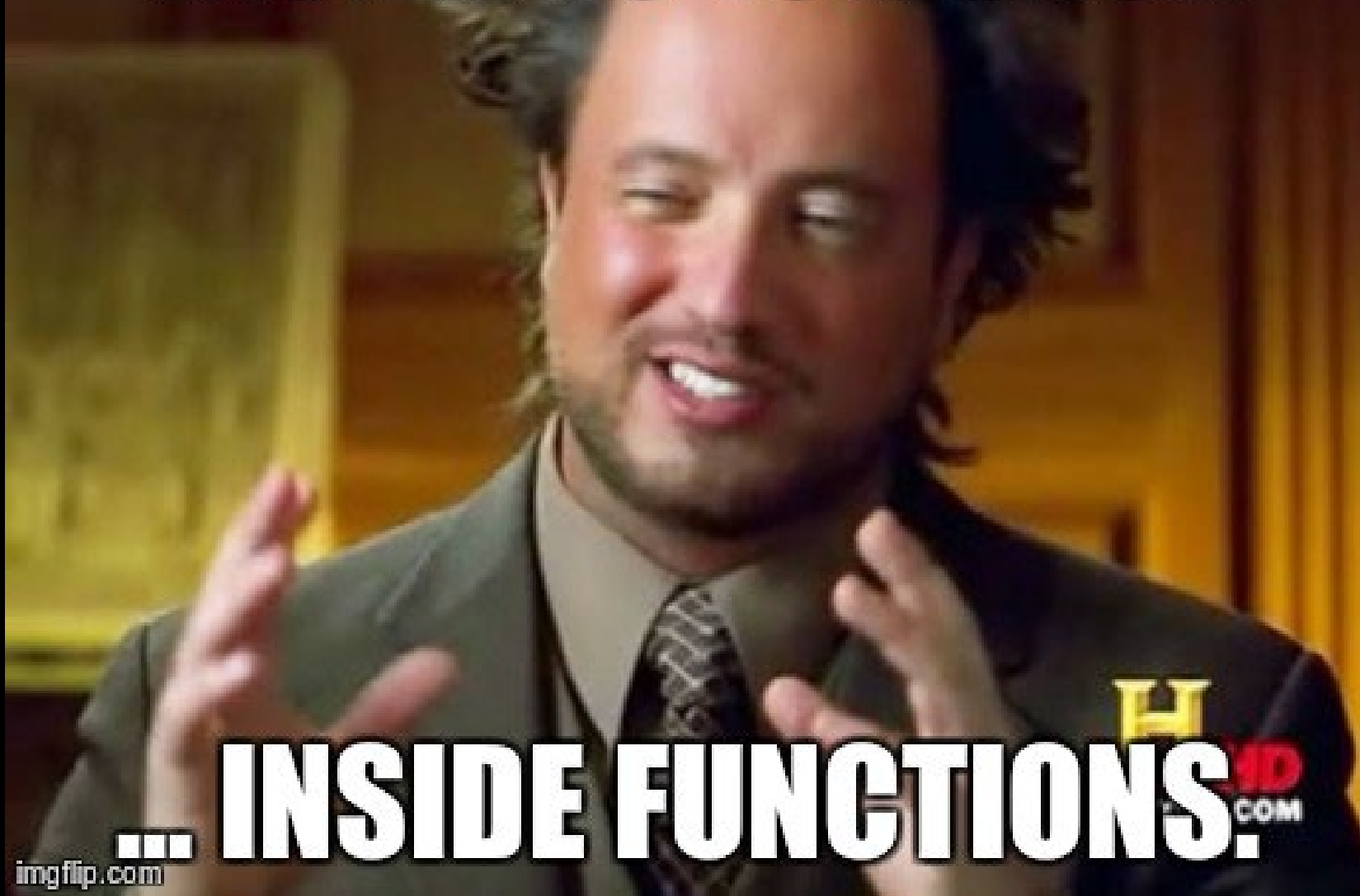


**Bygga ut vårt program  
ännu mer!**

Nu med snyggare veckomeny!

```
def todays_lunch(day, lunch):  
    print("Lunch " + day + " is: " + lunch)  
  
def heading(message):  
    print("="*40)  
    print(message)  
    print("="*40)  
  
def main():  
    heading("Welcome!")  
    todays_lunch("Monday", "Spagetti Bolognaise")  
    todays_lunch("Tuesday", "Fish and chips")  
    todays_lunch("Wednesday", "Hamburger")  
    todays_lunch("Thursday", "Soup")  
    todays_lunch("Friday", "Pizza")  
    heading("Goodbye!")  
  
main()
```

**I'VE GOT FUNCTIONS  
INSIDE FUNCTIONS...**

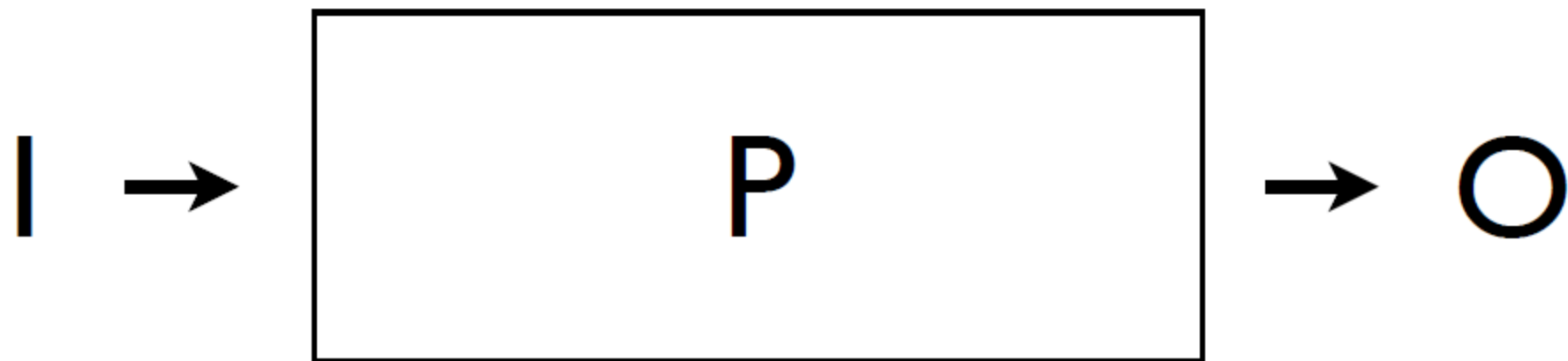


**... INSIDE FUNCTIONS.**

# Sammanfattning såhär långt

- **Vi bygger funktioner för att dela upp vår kod**
  - Varje funktion har en specifik uppgift
  - Varje funktion skapas genom nyckelordet "def"
- **Vi kan skicka med information till våra funktioner för att skräddarsy dem**
  - Det kallas parametrar där man skapar funktionen
  - Det kallas argument där man kör (kallar på) funktionen
- *Viktigt! Vad händer om man har två funktioner med samma namn?*

Att **returnera** värde från en  
funktion



32

str

"32"



# Funktioner med returvärdet

- Ofta vill man att funktion ska göra en specifik sak – och sedan returnera (skicka tillbaka) värdet som genereras av funktionen. Detta används ofta vid:
  - Beräkningar
  - Användarinput
- Vi har redan använt sådana funktioner som returnerar värde, som t.ex.
  - `type()`
  - `input()`
  - `str()`

# Exempel - returvärdet

```
name = input("Hej! Vänligen ange ditt namn: ")
```

```
Hej! Vänligen ange ditt namn: Anton
```

```
print(name)
```

```
.
```

# Returvärden

- Funktionen genomför sina instruktioner – och avslutar med att returnera (skicka tillbaka) ett resultat.
- T.ex. en funktion som omvandlar meter till yards.
  - Parameter: meters
  - Returvärde: yards
- Skulle kunna se ut på följande sätt:

```
def meters_to_yards(meters):  
    yard_per_meter = 0.9144  
    result = meters/yard_per_meter  
    return result
```



```
print(meters_to_yards(100))  
print(meters_to_yards(200))  
print(meters_to_yards(300))
```

```
109.36132983377078  
218.72265966754156  
328.0839895013123
```

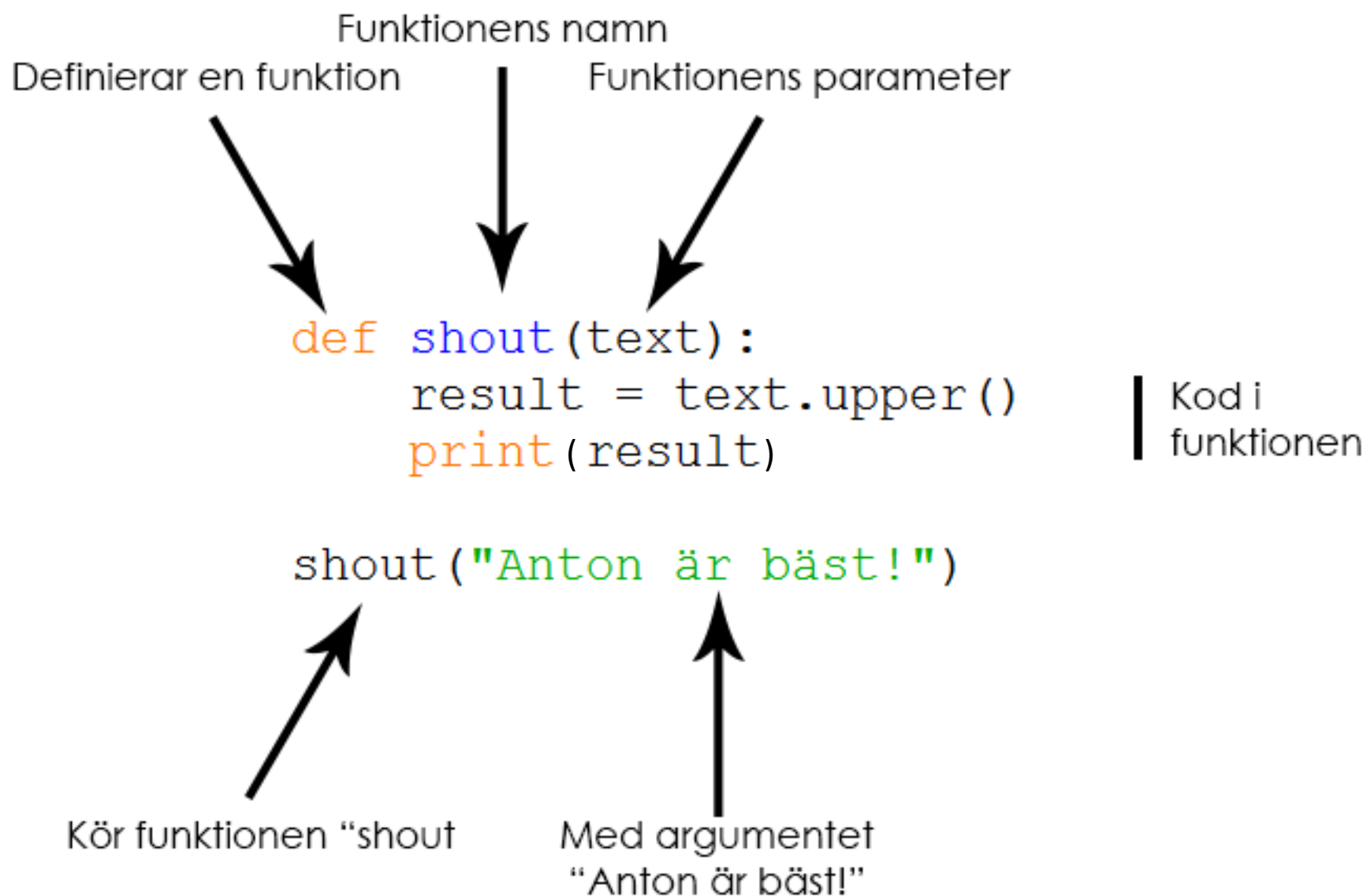
# DEMO - Returvärden

Hur kan vi snyggga till detta  
program?

A still from the movie Toy Story showing Woody and Buzz Lightyear. Woody is on the left, looking slightly concerned. Buzz is on the right, looking excited with his arms raised. The background is a blurred outdoor setting.

**FUNCTIONS**

**FUNCTIONS EVERYWHERE**



# Men, namngivning? Varför är det viktigt? Och varför är det svårt?

**A**

```
def shout(text):  
    result = text.upper()  
    print result  
  
shout("Anton är bäst!")
```

**C**

```
def hejsan(hoppsan):  
    tjosan = hoppsan.upper()  
    print tjosan  
  
hejsan("Anton är bäst!")
```

**B**

```
def a(b):  
    c = b.upper()  
    print c  
  
a("Anton är bäst!")
```