Introduktion till programmering

Funktioner

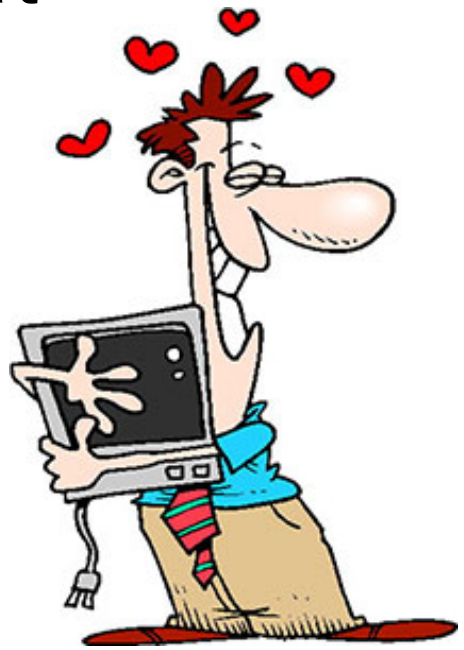
Dagens föreläsning

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 * 2
  - 20 / 4

```
>>> 5 + 5
10
>>> 10 * 2
20
>>> 20 / 4
5
```

Exempel på instruktioner

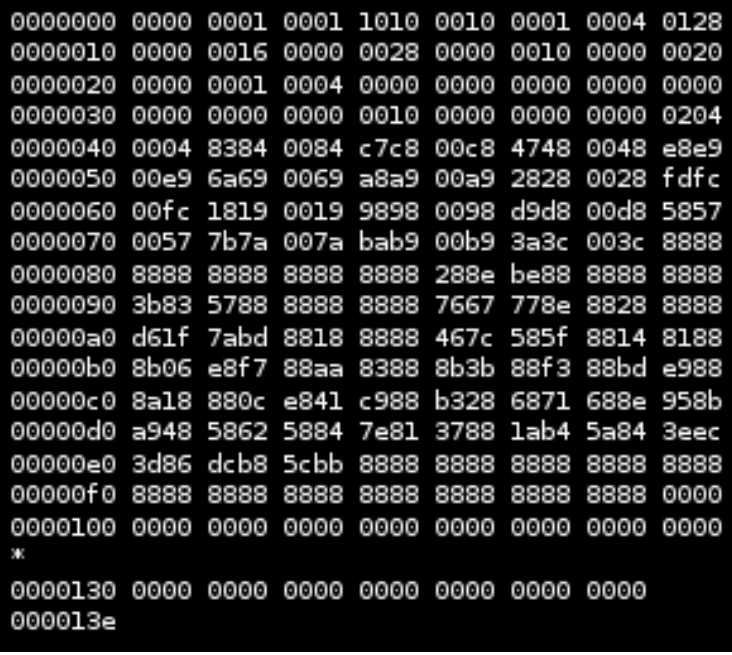
- Instruktioner att skriva ut saker (output)
  - print (”Hello World!”)
  - print (”Tjena kexet, sitter du här och smular?”)

```
>>> 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(”Hej, vad heter du?”)
  - input(”Vilket är Sveriges bästa fotbollslag?”)

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




```python
# 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

Toppsäljande kampanjprodukter

-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 kundvagn

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

HP 250 G6 + Ryggsäck Core i5 8GB 256GB 15.6

Kraftfull bärbar med högupplöst skärm och stor SSD-hårddisk. Just nu medföljer exklusiv ryggsäck från HP.

- Processortyp Core i5
- Processorgeneration 7
- Installerad minnesstorlek 8 GB
- Lagringskapacitet 256 GB

+100 kvar Lägg i kundvagn

-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 kundvagn

-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

+100 kvar Lägg i kundvagn

-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 pixels

-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

-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.

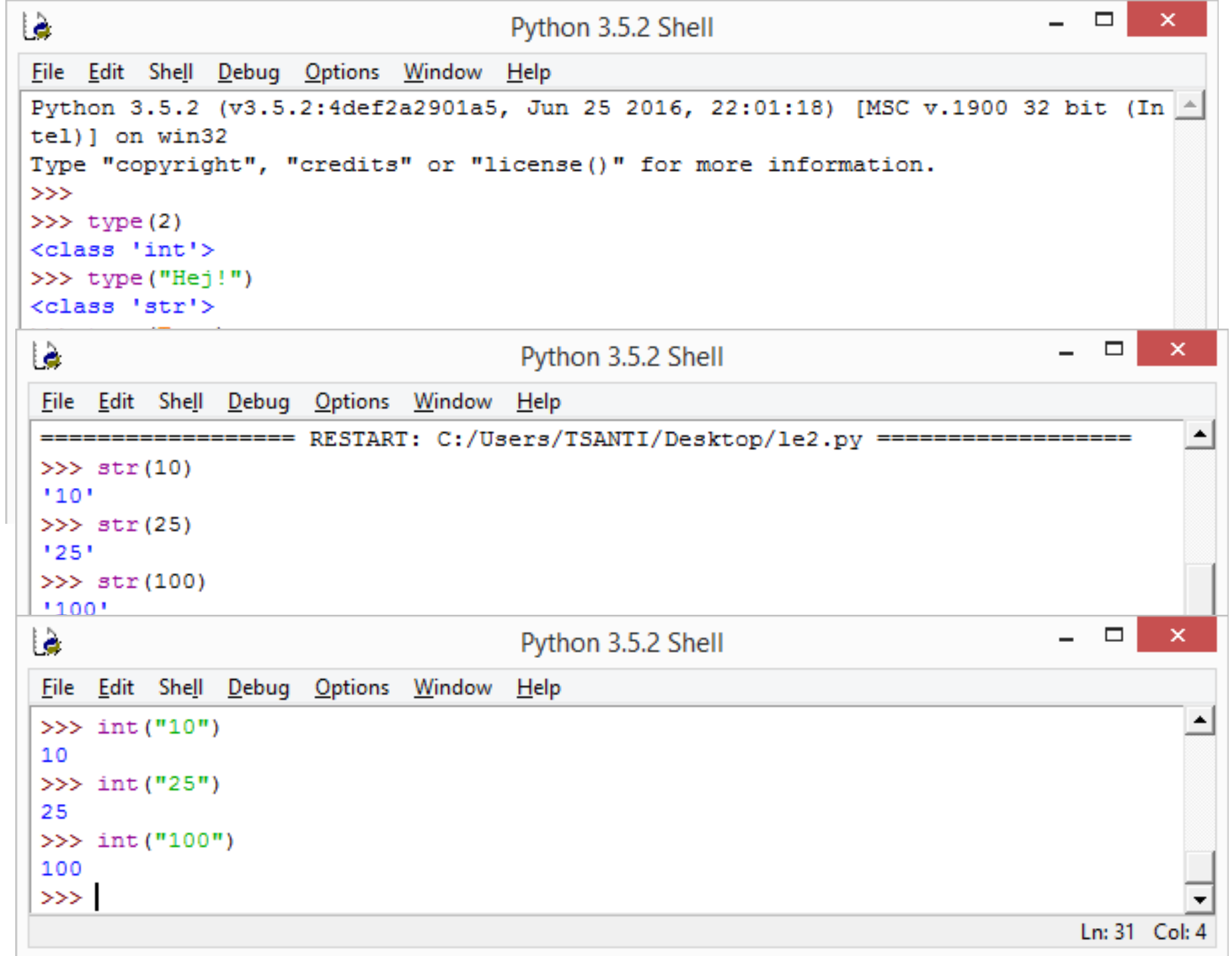
Inbyggda funktioner i Python

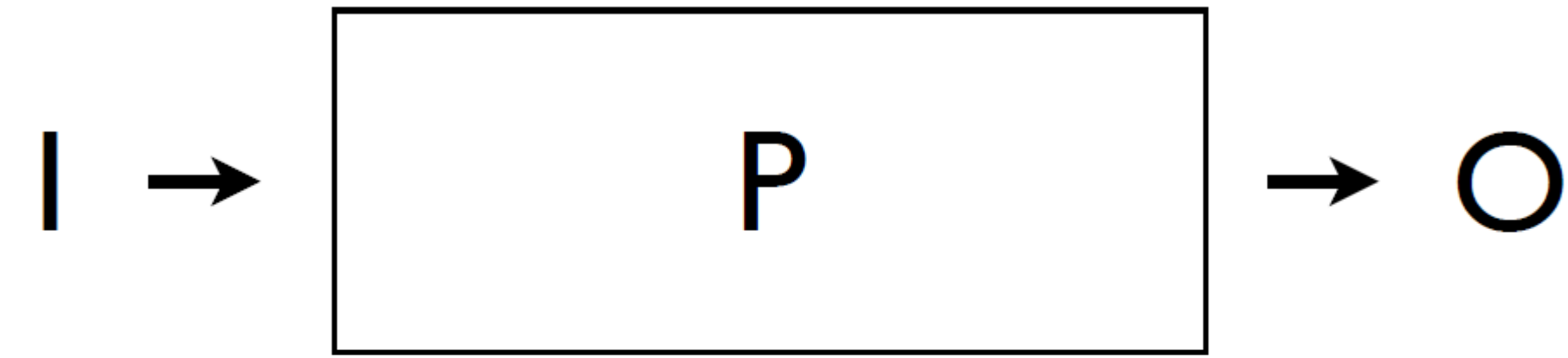
- Det finns många inbyggda funktioner i Python, vissa har vi redan använt. De är uppbyggda på samma sätt:

  - 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**

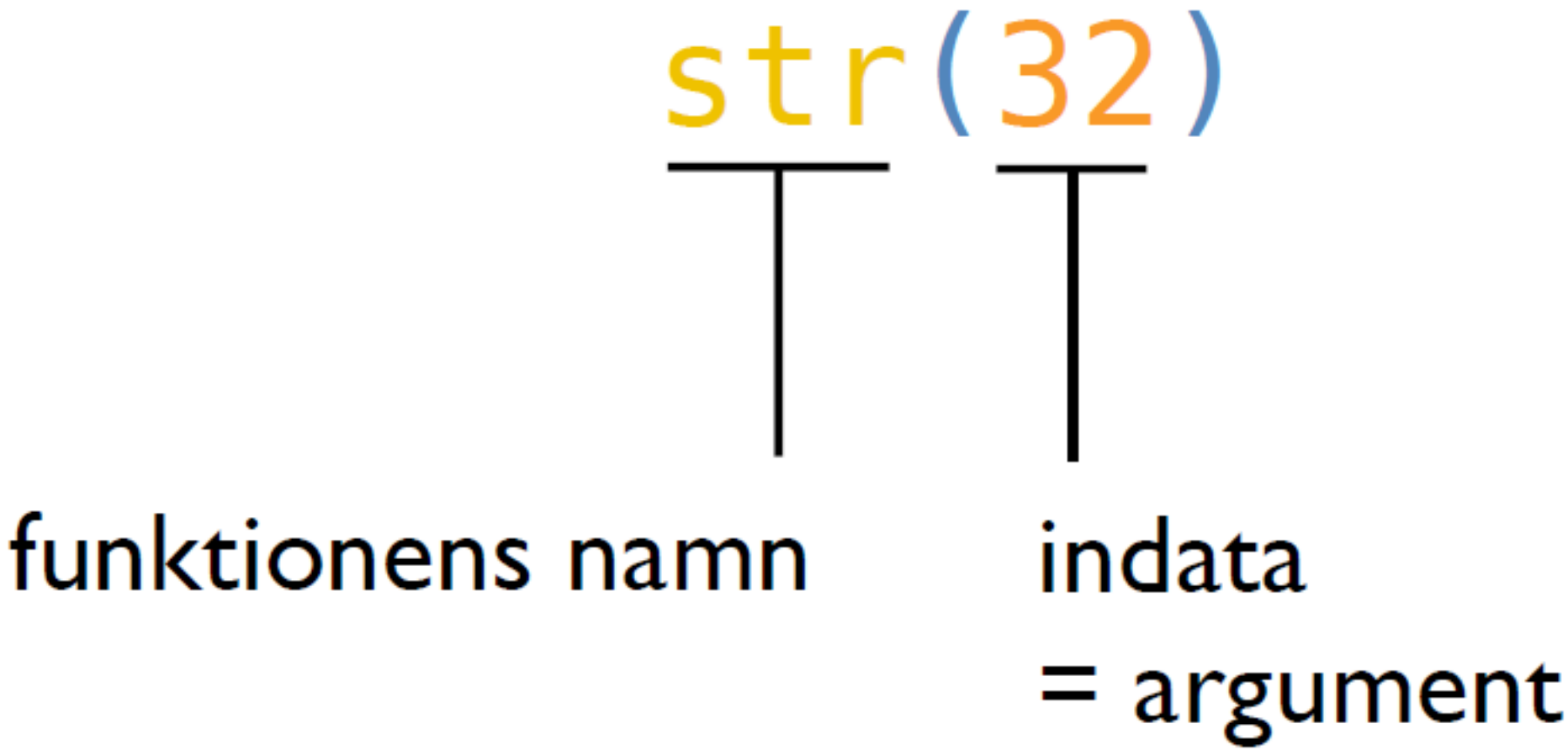

Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
>>> type(2)
<class 'int'>
>>> type("Hej!")
<class 'str'>
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
========== RESTART: C:/Users/TSANTI/Desktop/le2.py ==========
>>> str(10)
'10'
>>> str(25)
'25'
>>> str(100)
'100'
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> int("10")
10
>>> int("25")
25
>>> int("100")
100
>>>
Ln: 31 Col: 4



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

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/TSANTI/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

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

welcome()
```

JavaScript

```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:

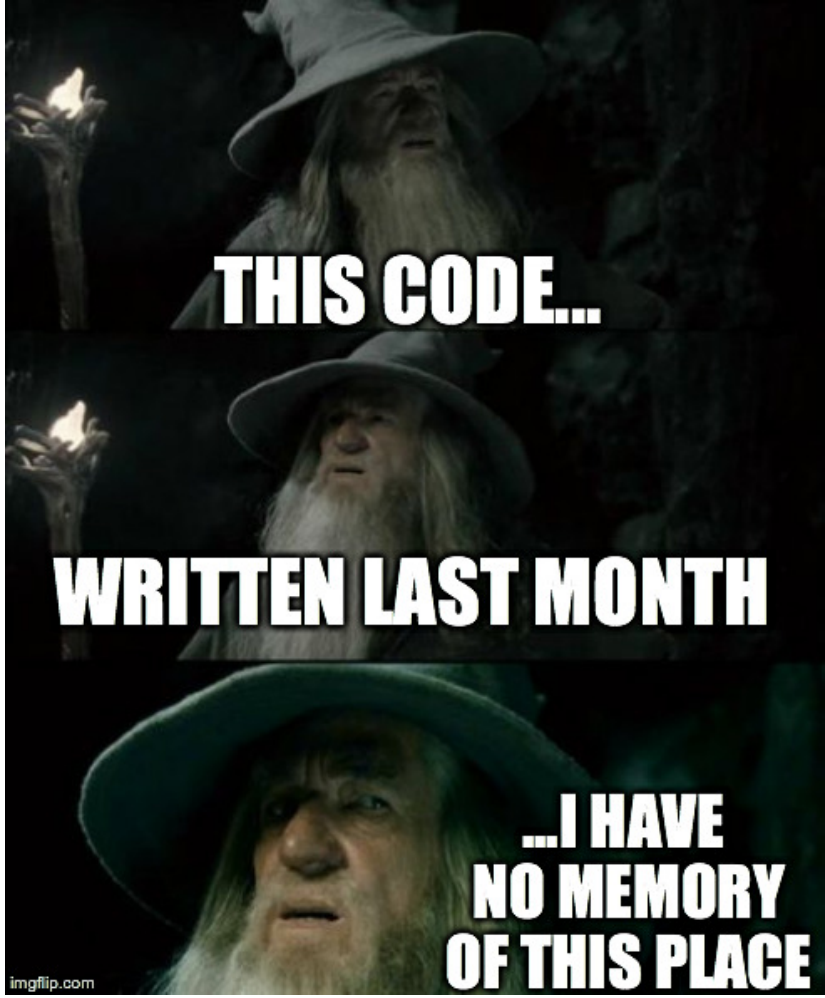
```
def welcome():
    print("-"*40)
    print("Welcome")
    print("-"*40)

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)

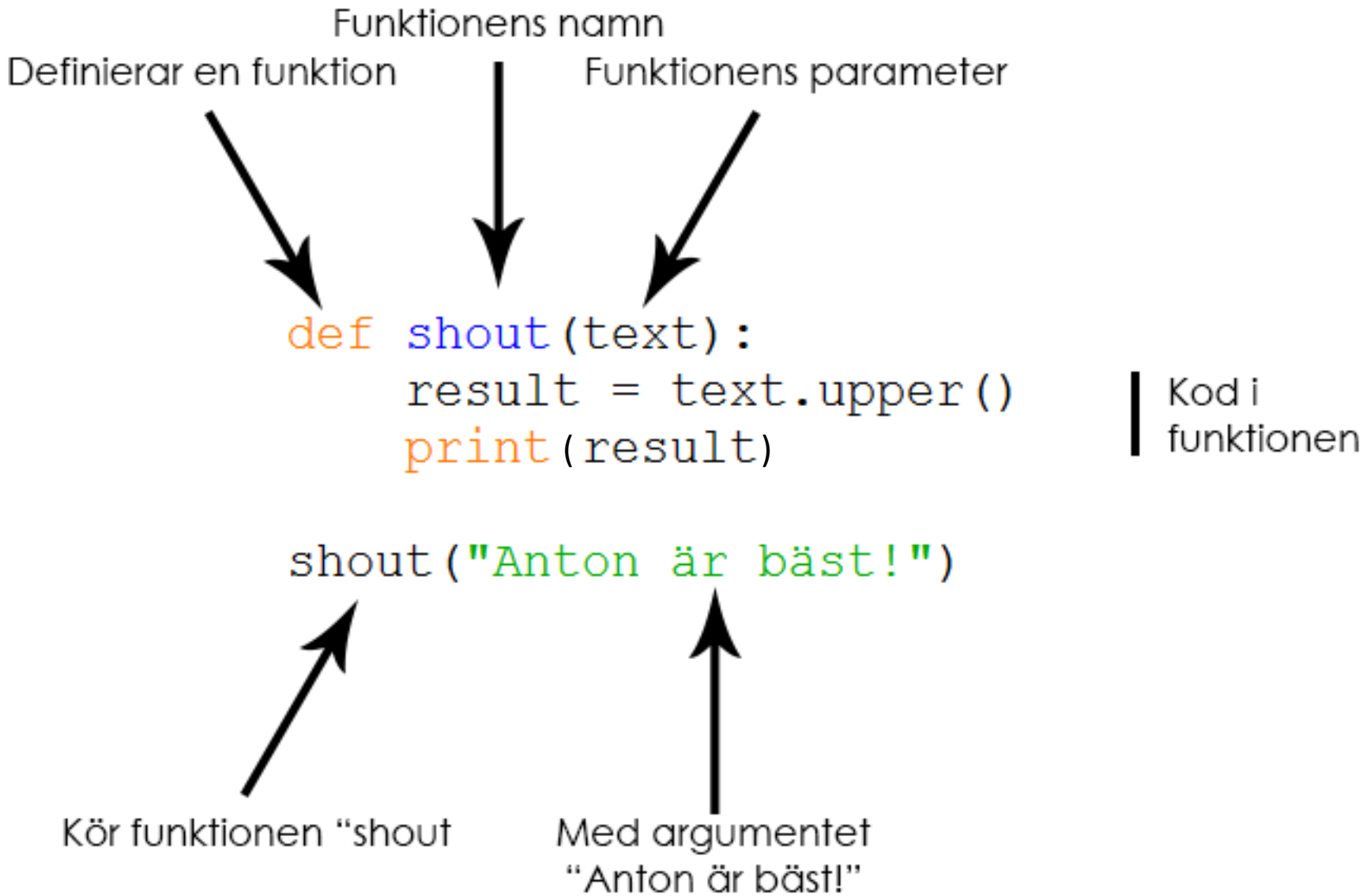
Dokumentation för funktioner

- 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
```


Funktionens namn
Definierar en funktion
Funktionens parameter
def shout(text):
result = text.upper()
print(result)
Kod i funktionen
shout("Anton är bäst!")
Kör funktionen “shout
Med argumentet “Anton är bäst!”

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!
==========================================
```

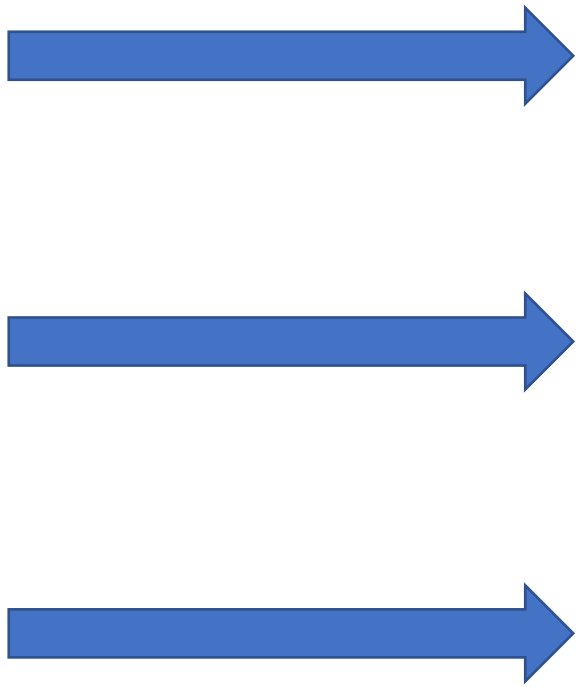
```
print("="*40)
print("Welcome!")
print("="*40)
print("Lunch Monday is: Spagetti Bolognaise")
print("Lunch Tuesday is: Fish and chips")
print("Lunch Wednesday is: Hamburger")
print("Lunch Thursday is: Soup")
print("Lunch Friday is: Pizza")
print("="*40)
print("Goodbye!")
print("="*40)
```

```
========================================
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

2

3



```
========================================

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!

========================================
```

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

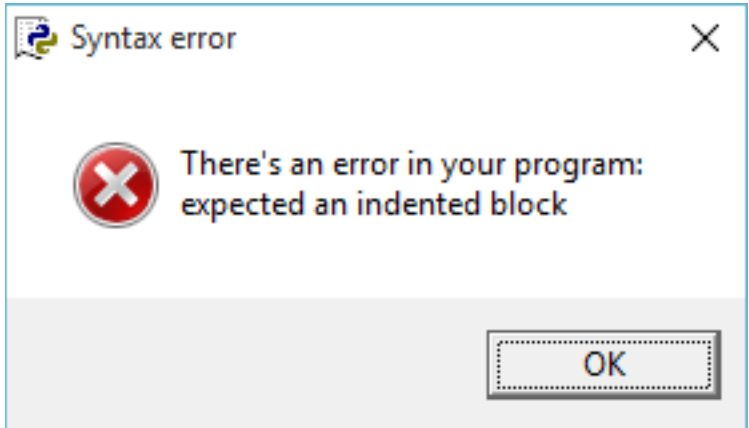
- Tänk på att koden läses *uppifrån och ner*. Detta innebär att vi kan inte kalla på en funktion innan den är deklarerad (skriven). Då får man detta fel:

```
welcome()

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

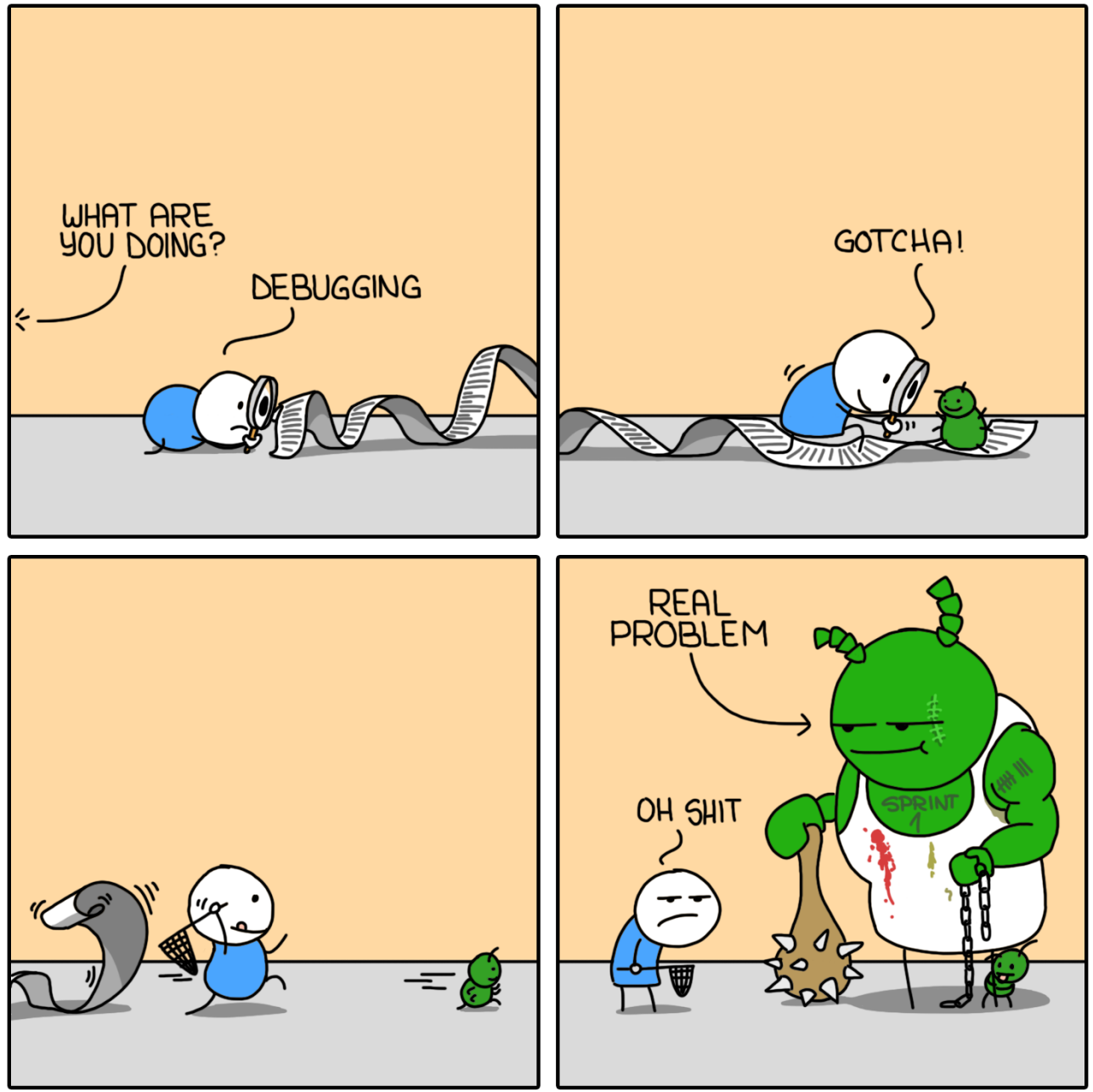
```
Traceback (most recent call last):
  File "C:\Users\Anton\Box Sync\HT 2015\DA354A - Introduktion till programmering
\Föreläsningar\Föreläsning 2 - Funktioner\Exempel\1-Funktioner.py", line 1, in <
module>
    welcome()
NameError: name 'welcome' is not defined
```

- 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

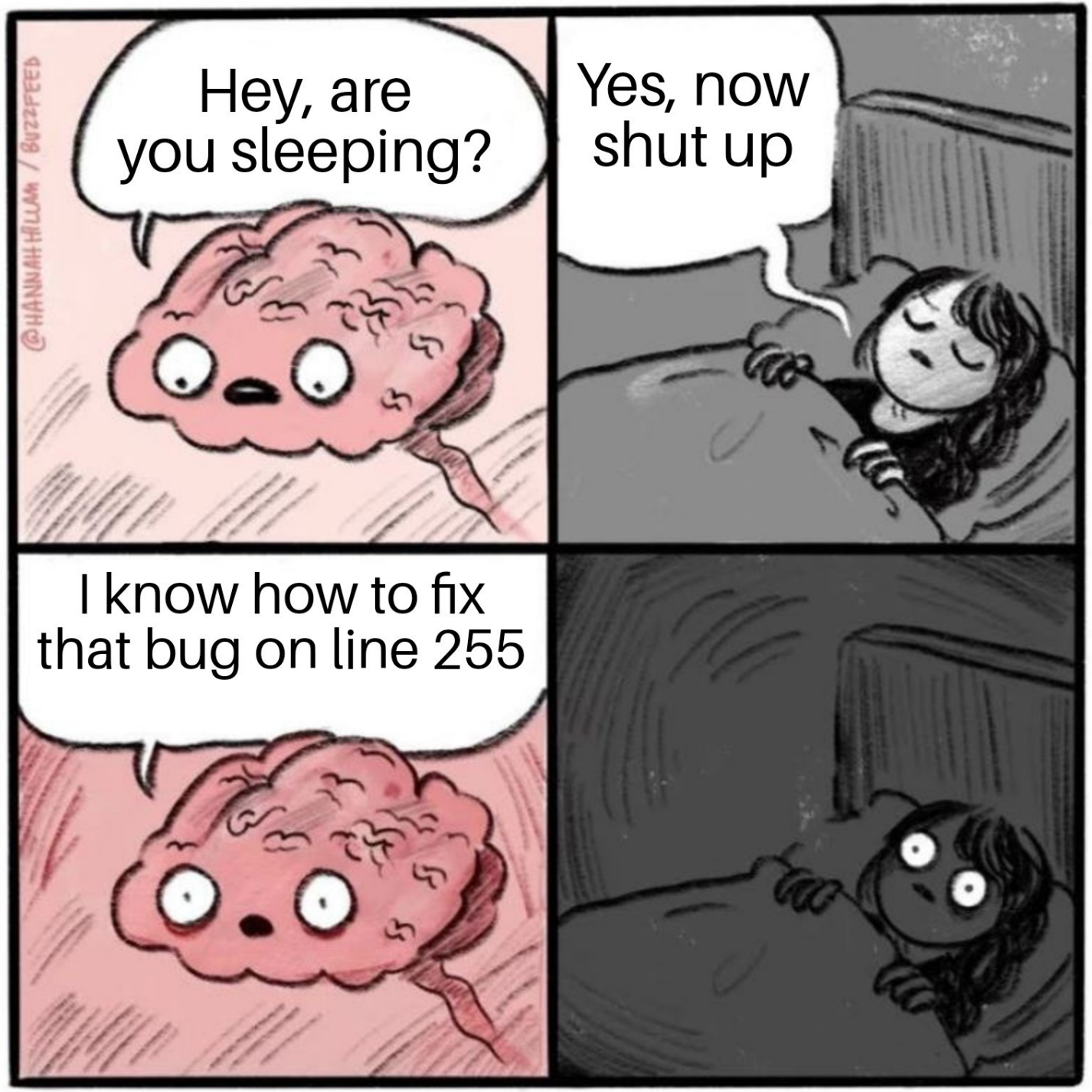


MONKEYUSER.COM

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
@HANNAHHILLAM / BUZZFEED

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.

Parameter

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

Argument

```
heading("Welcome!")
weeks_lunch()
heading("Goodbye!")
```

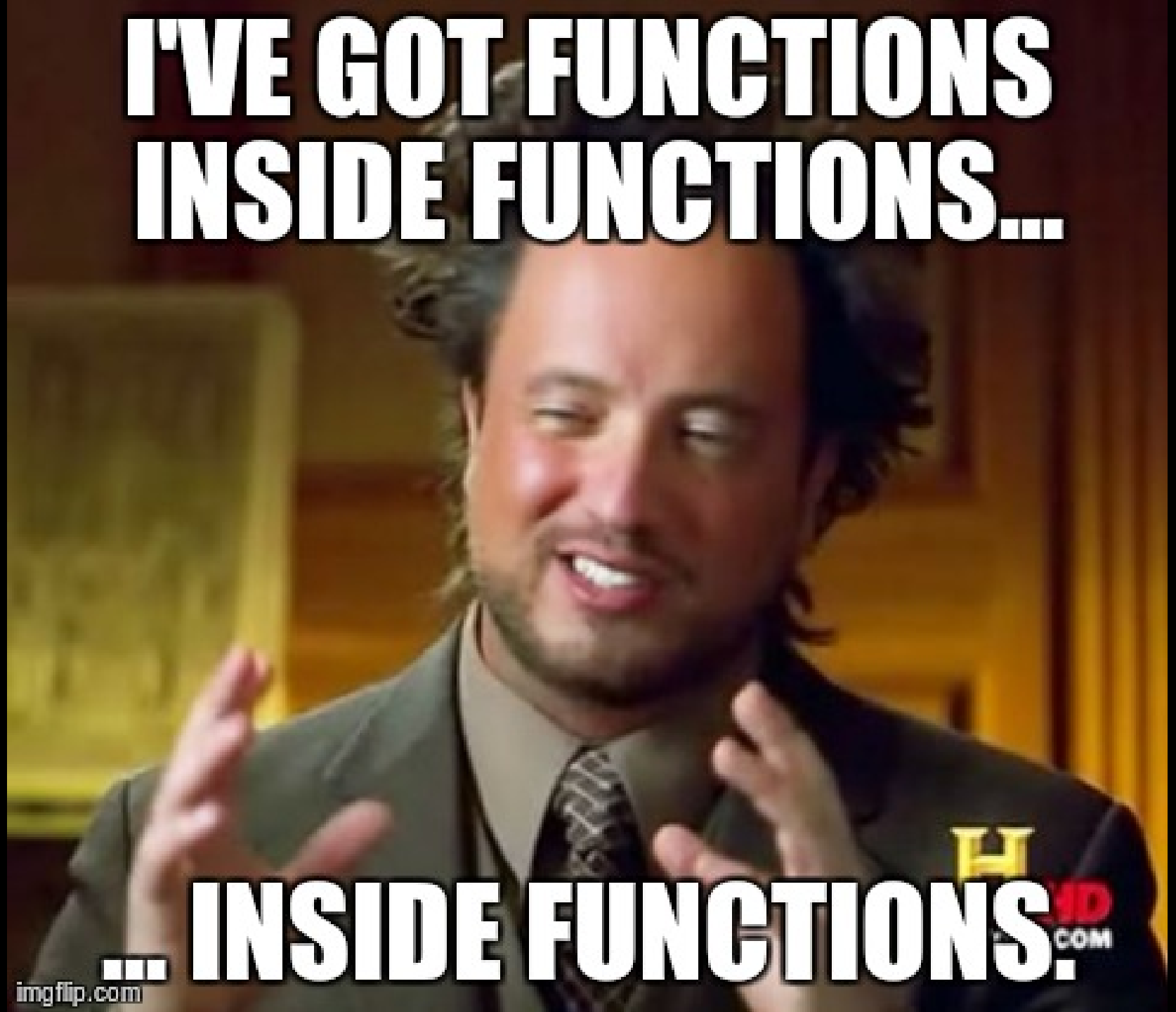
Argument

```
========================================
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!
========================================
```

Bygga ut vårt program ännu mer!

Nu med snyggare veckomeny!

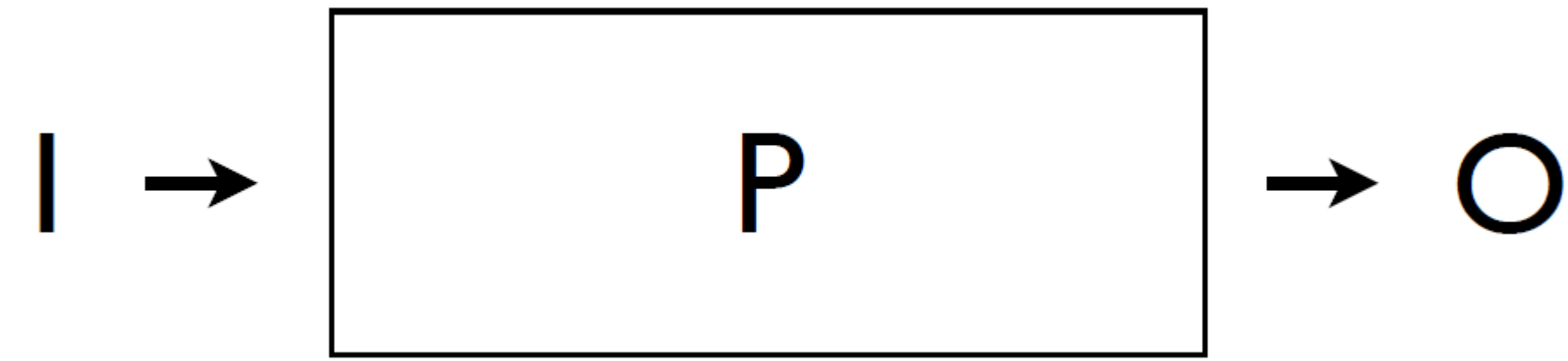
```
def todays_lunch(day, lunch):
    print("Lunch " + day + " is: " + lunch)

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.
imgflip.com

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 returvärden

- 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 - returvärden

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

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

```
print(name)
```

Returvärden

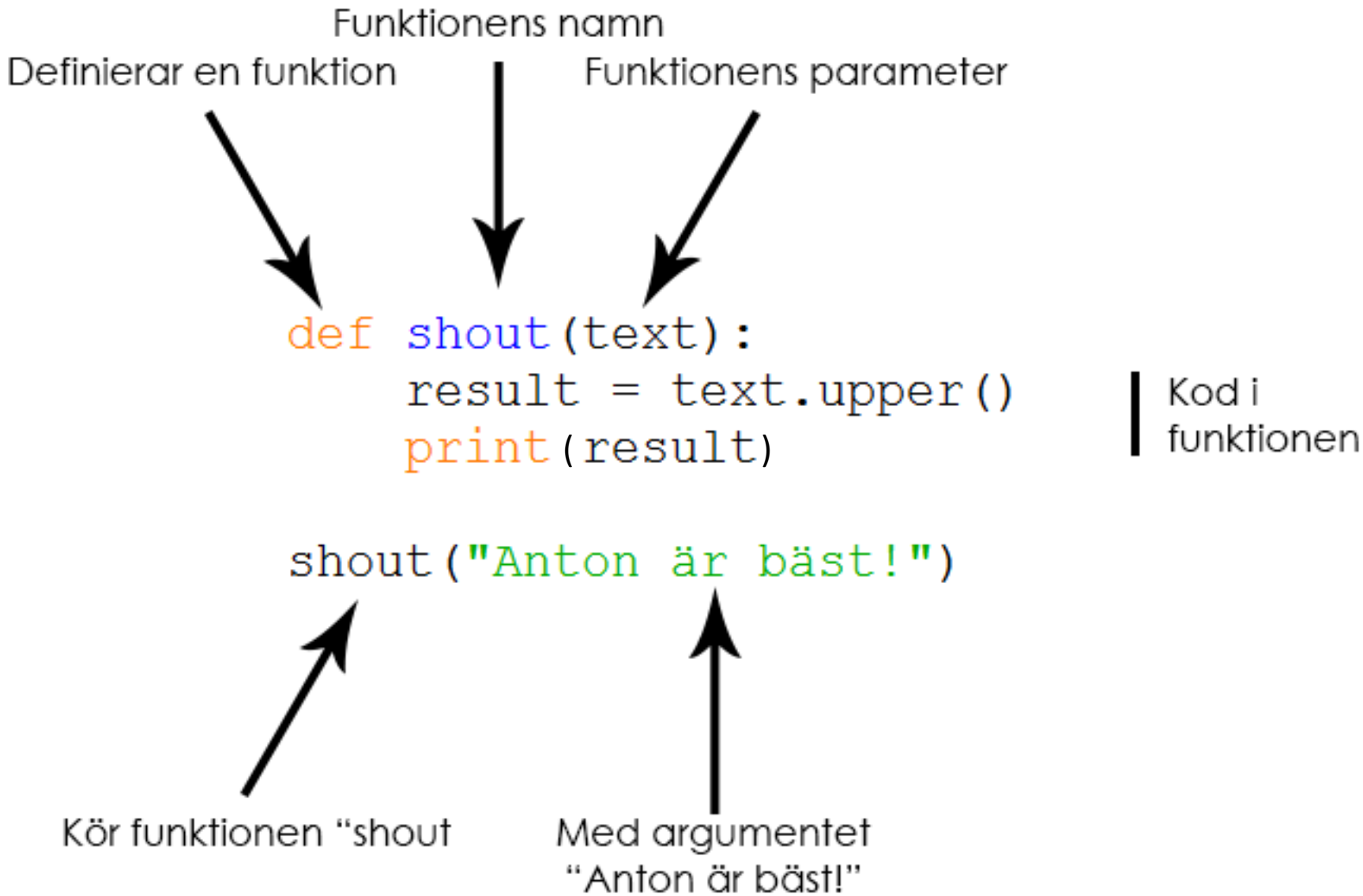
- Funktionen genomför sina instruktioner – och avslutar med att returnera (skicka tillbaka) ett resultat.
- T.ex. en funktion som omvandlar meter till yards.
  - Parameter: meters
  - Returvärde: yards
- Skulle kunna se ut på följande sätt:

```
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


FUNCTIONS
FUNCTIONS EVERYWHERE


Funktionens namn
Definierar en funktion
Funktionens parameter
def shout(text):
result = text.upper()
print(result)
Kod i funktionen
shout("Anton är bäst!")
Kör funktionen “shout
Med argumentet “Anton är bäst!”

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!")
```

Moduler

Inbyggda funktioner i Python

Moduler – Funktioner för användning

- https://docs.python.org/3/py-modindex.html
- Många moduler som vi kan använda oss utav, t.ex.
  - math
- Där finns funktioner som t.ex.
  - ceil Avrundar ett tal uppåt
  - floor Avrundar ett ta nedåt
  - isnan Kontrollerar om ett tal **inte** är ett nummer
  - pi Returnerar pi

En annan modul

- Som ni kommer att använda er utav på inlämningsuppgiften är:
  - Random

- *Random*-modulen har t.ex. följande funktioner

  - ***randint****(a, b)* Genererar ett slumptal mellan a & b (inkl, a & b)
  - ***sample****(population, k)* Väljer k antal värden från ett antal givna värden
  - ***random****()* Slumpar ett tal mellan 0,0 och 1,0

Att använda sig utav moduler

- Det är väldigt enkelt att använda sig utav dessa inbyggda moduler. Vill vi använda oss utav modulen ”math” skriver vi:

```
# Importerar alla funktioner från modulen "math"
from math import *
print(floor(4.321))
print(ceil(4.321))
```

```
4.0
5.0
```

```
# Importerar funktionerna "floor" & "ceil" from modulen "math"
from math import floor, ceil
print(floor(4.321))
print(ceil(4.321))
```

```
4.0
5.0
```

```
# Importererar modulen "math"
import math
print(math.floor(4.321))
print(math.ceil(4.321))
```

```
4.0
5.0
```

Demo för moduler

Egna moduler

- Vi kan skapa egna moduler
  - I vårt fall genom en samling av funktioner
- Modulerna är ett eget dokument, med funktioner inuti sig.
  - Man anger sökväg + filnamn (utan ”.py”) för at inkludera modulen.

4-Songlist.py - C:\Users\TSANTI\Box Sync\HT 2015\DA354A - Introduktion till pro...

File Edit Format Run Options Windows Help

```
import let_it_be

let_it_be.song()
```

let_it_be.py - C:\Users\TSANTI\Box Sync\HT 2015\DA354A - Introduktion till prog...

File Edit Format Run Options Windows Help

```
def song():
    vers_1()
    print ""
    vers_2()
    print ""
    chours()
    print ""
    vers_3()
    print ""
    vers_4()
    print ""
    chours_2()
    print ""
    chours()
    print ""
    chours()
    print ""
    vers_5()
    print ""
    vers_6()
    print ""
    chours_2()
    print ""
    chours()

def chours():
    print "Let it be, let it be"
    print "Let it be, let it be"
    print "Whisper words of wisdom"
    print "Let it be"

def chours_2():
    print "Let it be, let it be"
    print "Let it be, let it be"
    print "Yeah, there will be an answer let it be"

def vers_1():
    print "When I find myself in times of trouble"
    print "Mother Mary comes to me"
    print "Speaking words of wisdom, let it be"

def vers_2():
    print "And in my hour of darkness"
    print "She is standing right in front of me"
    print "Speaking words of wisdom, let it be"

def vers_3():
    print "And when all the brokenhearted people"
    print "Living in the world agree"
    print "There will be an answer, let it be"

def vers_4():
    print "For though they may be parted"
    print "There is still a chance that they will see"
    print "There will be an answer, let it be"

def vers_5():
    print "And when the night is cloudy"
```

Ln: 1 Col: 0

Demo egna moduler

Att skriva ut saker

Hur gör vi det snyggast?

```
name = "Anton"
age = 31
city = "Lund"

print(name + " är " + str(age) + "år och bor i " + city)
#=> Anton är 31år och bor i Lund
```

```
name = "Anton"
age = 31
city = "Lund"

print("{} är {}år och bor i {}".format(name, age, city))
#=> Anton är 31år och bor i Lund
```

```
name = "Anton"
age = 31
city = "Lund"

print(f"{name} är {age}år och bor i {city}")
#=> Anton är 31år och bor i Lund
```

```python
name = "Anton"
age = 31
city = "Lund"

print("{} är {}år och bor i {}".format(name, age, city))
#=> Anton är 31år och bor i Lund
```

Extratillfällen?

Inlämningsuppgiften!