

# Introduktion till programmering

## Fel- och filhantering

# Dagens upplägg

- Förbättrad felhantering
  - När allt inte går som planerat: try/except
- Filhantering
  - Öppna filer
  - Skapa filer
  - Läsa/redigera/ersätta innehåll i filer
  - Stänga filer
- Kombinerat exempel

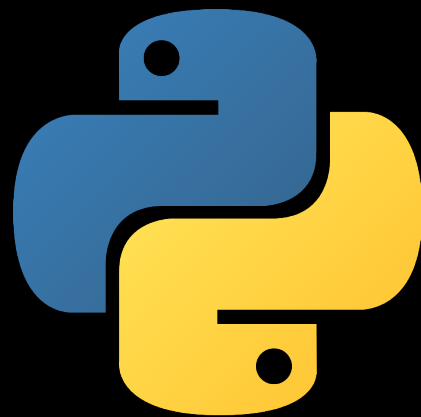




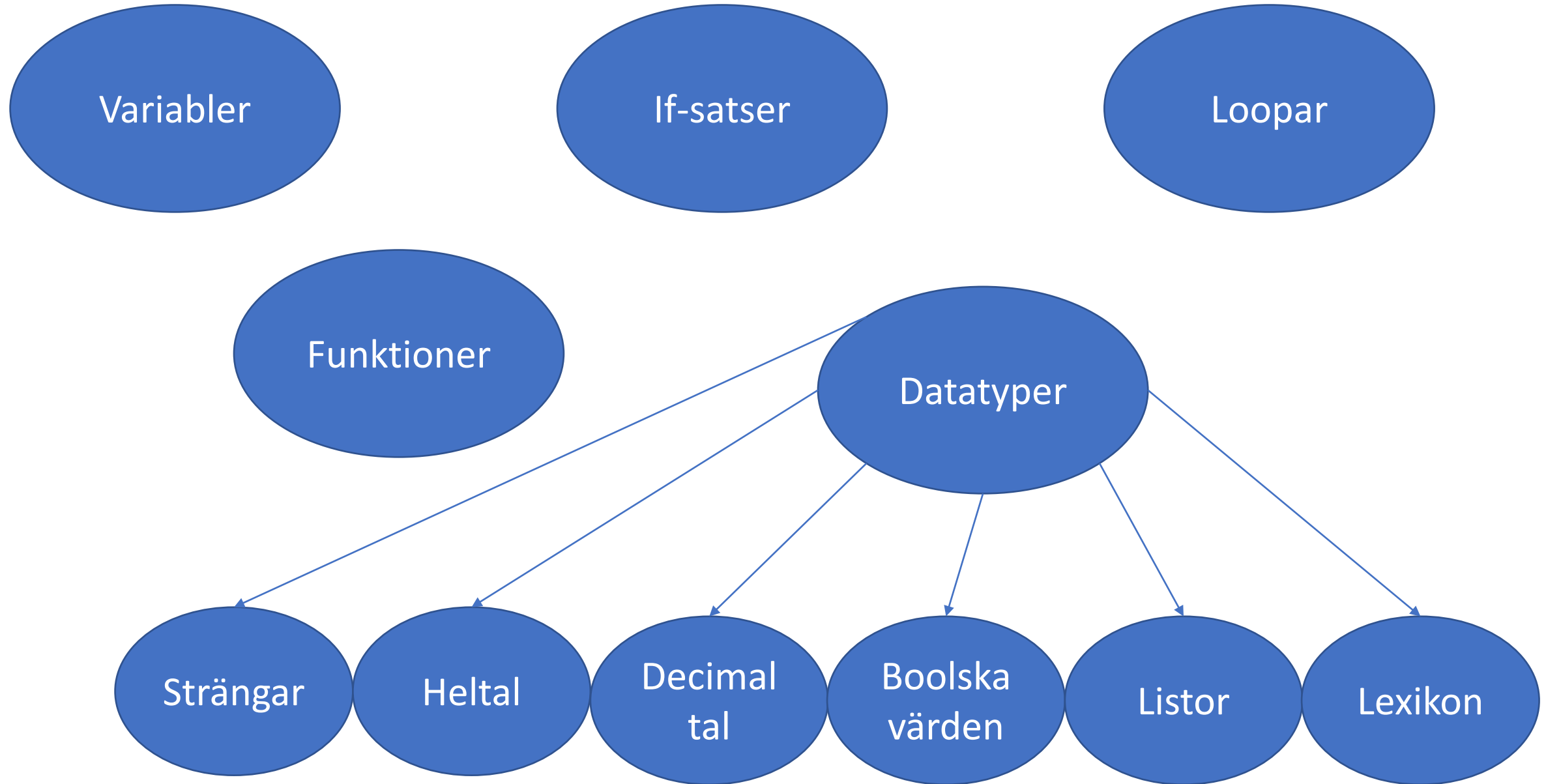
Några frågor?

47	21/11	13-15	F: If-satser & Loopar	Anton (inspelad föreläsning)	NI:A0406	3
	22/11	08-10	F: If-satser & Loopar (2)	Anton	NI:A0406	
	23/11	08-12	L: If-satser & Loopar	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	
48	27/11	10-12	F: Listor & Lexikon	Anton	NI:A0306	4
	28/11	13-15	F: Listor & Lexikon (2)	Anton	NI:A0307	
	30/11	08-12	L: Listor & Lexikon	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	
49	4/12	10-12	F: Fil- & Felhantering	Anton	NI:A0406	5
	6/12	08-10 + 10-12	F: Fil- & Felhantering (2) + Labb	Anton	NI:A0307 + OR:D326 & OR:D377	
	7/12	08-12	L: Fil- & Felhantering	Hussan	NI:A0314, NI:A0318, NI:B0314	
50	14/12	08-12	L: Fil- & Felhantering, m.m.	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	6
	14/12	13-15	F: Python & Webben	Anton	NI:C0E11	
51	19/12	13-17	L: Python & Webben	Anton	NI:A0406	7
2	9/1	13-15	F: Objektorienterad programmering	Anton	NI:A0306	
	11/1	08-12	L: Objektorienterad programmering	Anton (digitalt)	NI:A0314, NI:A0318, NI:B0314	

- Modul 4: Inlämningsuppgift 3 har deadline **onsdagen den 6/12** (detta är en gruppuppgift)
- Modul 5: Inlämningsuppgift 4 har deadline **söndagen den 17/12** (OBS. söndag)
- Modul 6: Inlämningsuppgift 5 har deadline **söndagen den 14/1** (OBS. söndag)

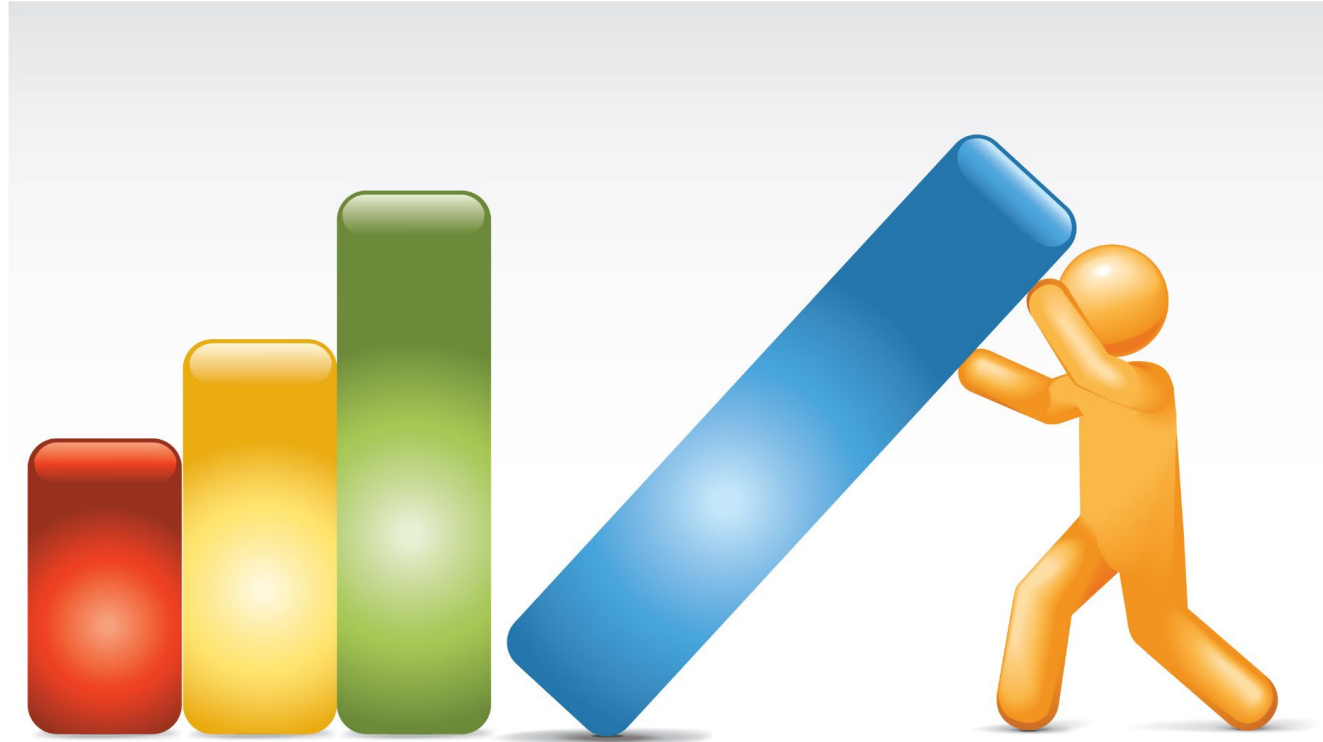


Vad har vi gjort hittills?



# Vi har lärt oss att göra program!

och att programmera!



**HELL**

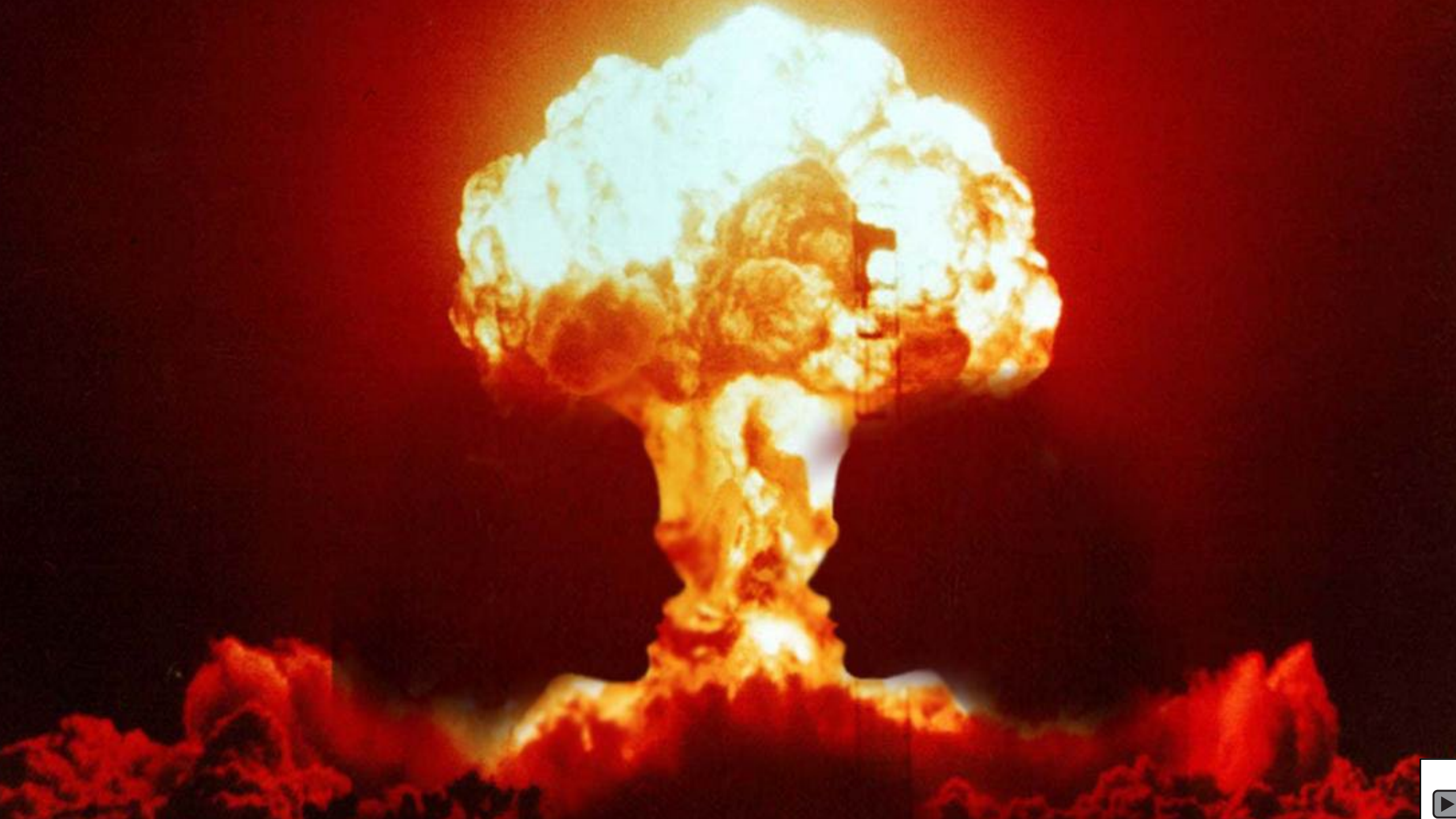


**YEAH**



# Programkrascher

Ni har nog varit med om det?

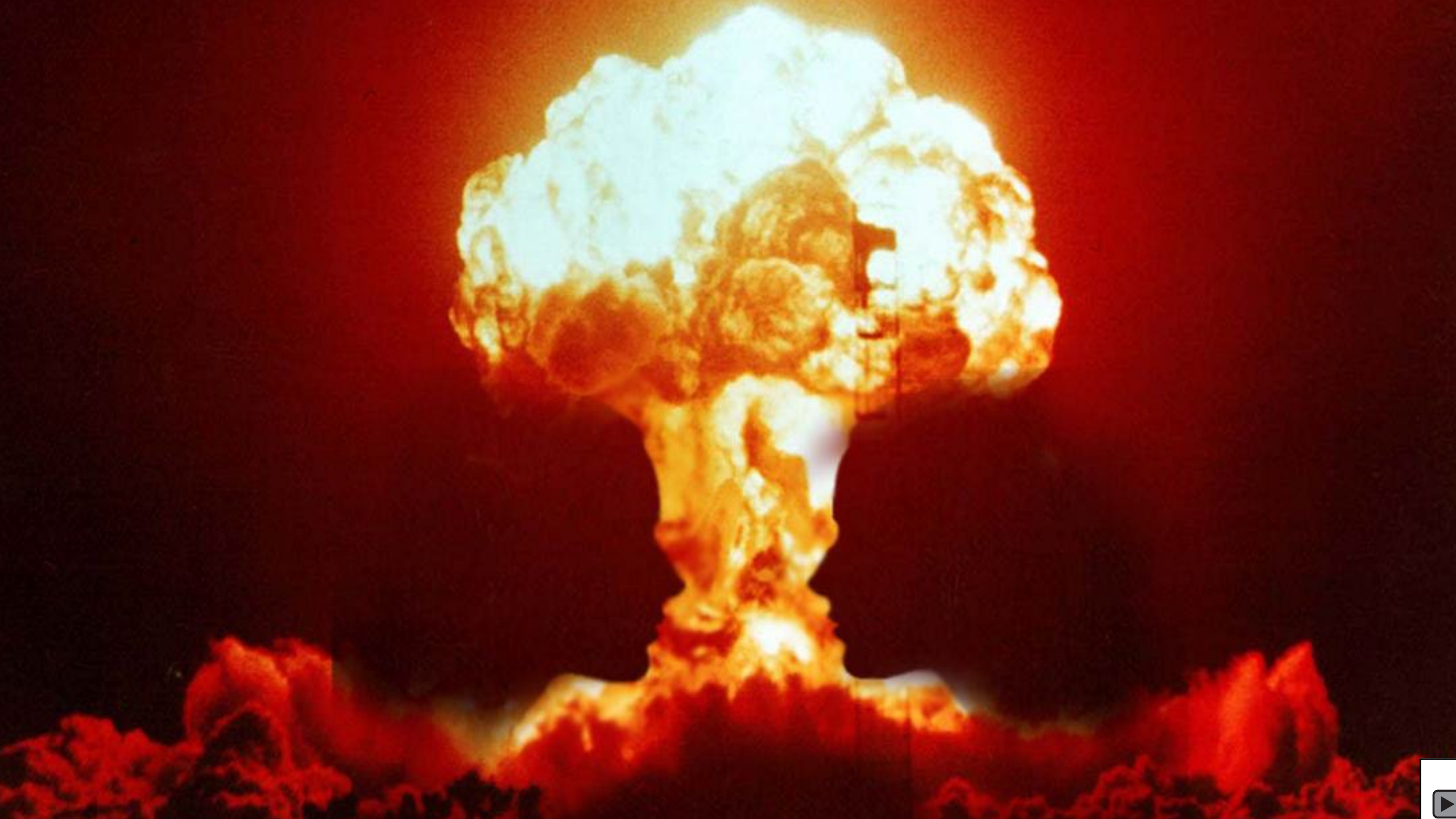


```
>>> print(name)
```

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

```
  File "<pyshell#3>", line 1, in <module>  
    print(name)
```

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



```
>>> print(name)
```

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

```
  File "<pyshell#3>", line 1, in <module>  
    print(name)
```

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

# När får vi fel?

När brukar era fel dyka upp?

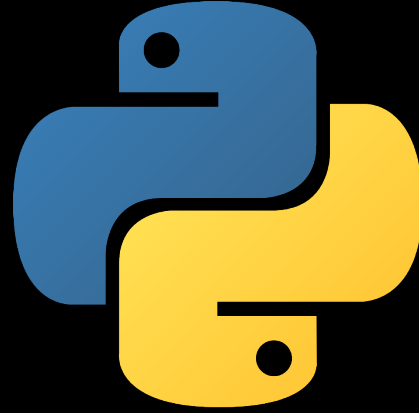


## How users see developers



## How developers see users





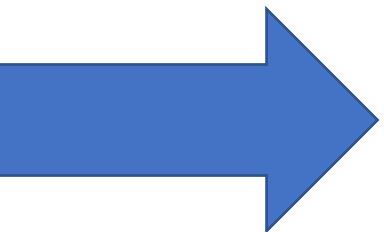
# Designa för fel.

Alla gör fel, förr eller senare. Tänk användbarhet.



```
movies = ["Star Wars", "Fight Club", "Titanic"]  
index = int(input("Ange index: "))  
print(movies[index])
```

Ange index: 3




```
Traceback (most recent call last):  
  File "C:/Users/TSANTI/Desktop/try_except_files.py", line 3, in <module>  
    print(movies[index])  
IndexError: list index out of range
```

```
movies = ["Star Wars", "Fight Club", "Titanic"]  
index = int(input("Ange index: "))  
print(movies[index])
```

Ange index: Hej

```
Traceback (most recent call last):  
  File "C:/Users/TSANTI/Desktop/try_except_files.py", line 2, in <module>  
    index = int(input("Ange index: "))  
ValueError: invalid literal for int() with base 10: 'Hej'
```



**This did not**

**go as planned**



Det blev ett *undantag*

Ett undantag (eng. exception) är ett fel som uppstår medan ett program körs, vilket medför att programmet abrupt stoppas (kraschar).

Du kan använda **try** / **except** för att snyggt hantera dessa fel/undantag.

[https://www.tutorialspoint.com/  
python3/python\\_exceptions.htm](https://www.tutorialspoint.com/python3/python_exceptions.htm)

Lista på undantag

# Hantera fel – fel sker alltid

- I Python kan man använda **try** för försöka utföra något
- Skulle det man försöker utföra gå fel, kan man fånga upp felet genom **except**

```
try:  
    # Några kodrader  
except:  
    # Blir något fel – kör denna kod
```

# Python kan ge oss olika typer av fel

- Exempel på typer av fel:
  - **NameError** När en variabel/funktion inte finns
  - **TypeError** När vi använder datatyper felaktigt
  - **IndexError** Vi försöker hämta ut ett värde från en lista genom ett index som inte finns
  - **ZeroDivisionError** Vi försöker dividera med 0
  - **ImportError** Vi försöker importera något som inte finns
  - Osv.
- Fler typer av fel hittar ni här:  
<https://docs.python.org/3/library/exceptions.html>



**Hur hanterar vi  
undantagen?**

```
movies = ["Star Wars", "Fight Club", "Titanic"]
index = int(input("Ange index: "))

try:
    print(movies[index])
except IndexError:
    print("Det finns ingen film med det index!")
```

```
>>> ===== RESTART
```

```
>>>
```

```
Ange index: 2
```

```
Titanic
```

```
>>> ===== RESTART
```

```
>>>
```

```
Ange index: 10
```

```
Det finns ingen film med det index!
```

```
movies = ["Star Wars", "Fight Club", "Titanic"]
index = int(input("Ange index: "))

try:
    print(movies[index])
except IndexError:
    print("Det finns ingen film med det index!")
```

```
===== RESTART: C:/Users/TSANTI/Desktop/try_except_files.py
```

```
Ange index: Hej
```

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

```
  File "C:/Users/TSANTI/Desktop/try_except_files.py", line 2, in
```

```
    index = int(input("Ange index: "))
```

```
ValueError: invalid literal for int() with base 10: 'Hej'
```

```
>>> |
```

# Generella fel vs. Specifika fel

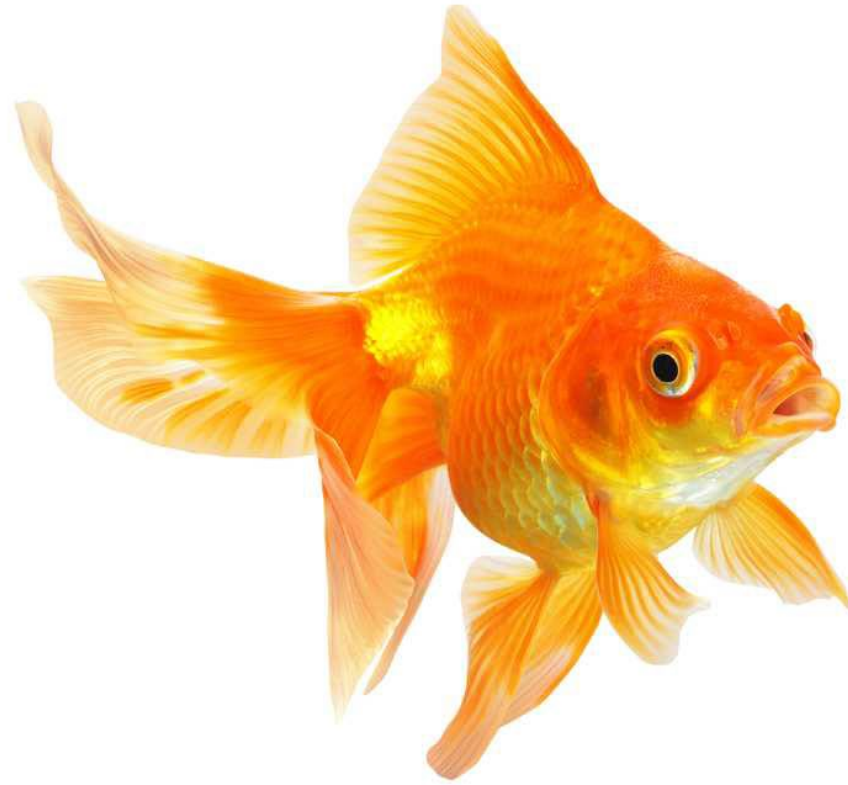
- Man kan i Python som vi sett, antingen "upptäcka":
  - Generella fel (alla fel)
  - Specifika fel (av en viss typ)
- Ibland vill man helgradera sig mellan flera fel, t.ex.

```
try:  
    # Några kodrader  
except IndexError:  
    # Index-fel  
except TypeError:  
    # Typ-fel
```

# Frågor på felhantering?

Genom try / except

Demo! =)



# Filhantering

Skapa program som kommer ihåg saker!

# Att spara text i filer

- Hittills har våra program varit helt nollställda när vi startat dem, vilket ibland har fungerat bra – men det vore ju roligt om vi kunde spara information mellan olika körningar.
- Det gör man enkelt genom att spara information i textfiler. Detta t.ex. genom
  - Ren text
  - Semikolonseparerade värden
  - Andra typer av strukturer, t.ex.
    - JSON
    - Pickle
    - XML
    - etc.



# Att öppna filer i Python

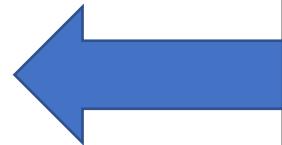
Öppnar i läsläge



```
my_file = open("demo.txt", "r")
```

```
my_file = open("demo.txt", "w")
```

Öppnar i skrivläge  
(ersätter innehåll)



```
my_file = open("demo.txt", "a")
```



Öppnar i skrivläge (lägger till innehåll)

r = read

w = write

a = append

# Funktioner för filer

- `read()` Returnerar all text i filen
- `readline()` Returnerar en rad åt gången
- `readlines()` Returnerar alla rader som en lista
- `write()` Skriver till filen

# Att läsa från en fil

```
my_file = open("demo.txt", "r")
content = my_file.read()
print("Fil: {}".format(my_file.name))
print("Innehåll: {}".format(content))
```

# Läsa in varje rad från en fil

```
my_file = open("demo.txt", "r")
print("Fil: {}".format(my_file.name))
print("Rad 1: {}".format(my_file.readline()))
print("Rad 2: {}".format(my_file.readline()))
print("Rad 3: {}".format(my_file.readline()))
```

# close()

Stänger filen när vi jobbat klart med den

Demo! =)