# TDP00 - Standardbibliotek

Pontus Haglund

Department of Computer and information science

LINKÖPING UNIVERSITY

1 Standardbibliotek
2 Moduler
   Spara data mellan körningar
3 Kommandoraden
4 Smått och gott
   Http
   Loggning
5 Undantag

LINKÖPING UNIVERSITY

# Vad är pythons standardbibliotek?

- https://docs.python.org/3/library/
- Egentligen allting som ’skeppas’ med python
- I detta sammanhang extra grejer vi kan importera

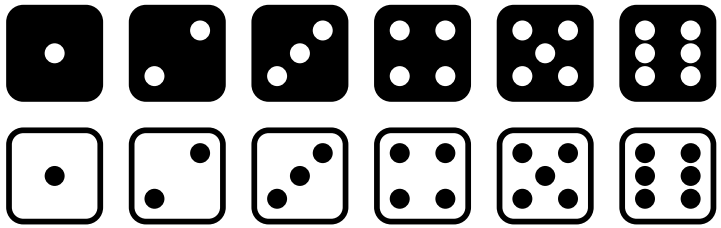
# Lyfta in moduler

```
import modulnamn
from modulnamn import funktion
from modulnamn import *
```

# import random

- importerar modulen
- åtkomst med `modulnamn.funktion`

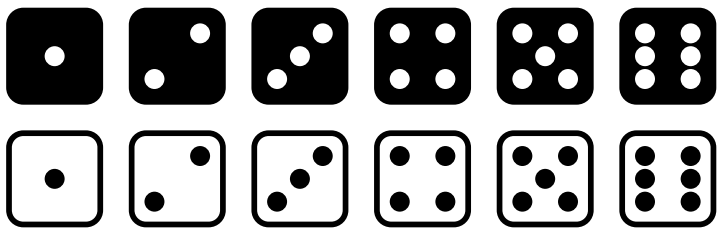
```
import random
print(random.randint(0, 5))
print(random.randrange(0, 5))
```



# from random import randint

- importerar den specificerade funktionen från modulen
- åtkomst med `funktion`

```
from random import randint
print(randint(0, 5))
print(randrange(0,5)) #fel varför?
```



# * är alla

- stjärna `expanderar` till alla funktionerna i modulen (nästan)
  - inledande understreck i namnet ger privata funktioner i moduler
- varför inte alltid allt?

```
from random import *
print(randint(0, 5))
print(randrange(0,5))
```

1 Standardbibliotek

2 Moduler

Spara data mellan körningar

3 Kommandoraden

4 Smått och gott

Http

Loggning

5 Undantag

LINKÖPING UNIVERSITY

# random

https://docs.python.org/3/library/random.html

- ibland vill man ha slumpat beteende

# random

Basic examples:

```
>>> random()                            # Random float:  0.0 <= x < 1.0
0.37444887175646646

>>> uniform(2.5, 10.0)                  # Random float:  2.5 <= x < 10.0
3.1800146073117523

>>> expovariate(1 / 5)                  # Interval between arrivals averaging 5 seconds
5.148957571865031

>>> randrange(10)                       # Integer from 0 to 9 inclusive
7

>>> randrange(0, 101, 2)                # Even integer from 0 to 100 inclusive
26

>>> choice(['win', 'lose', 'draw'])     # Single random element from a sequence
'draw'

>>> deck = 'ace two three four'.split()
>>> shuffle(deck)                       # Shuffle a list
>>> deck
['four', 'two', 'ace', 'three']

>>> sample([10, 20, 30, 40, 50], k=4)   # Four samples without replacement
[40, 10, 50, 30]
```

# datetime

https://docs.python.org/3/library/datetime.html

- hantering av datum, tid

]

# datetim

```
>>> import time
>>> from datetime import date
>>> today = date.today()
>>> today
datetime.date(2007, 12, 5)
>>> today == date.fromtimestamp(time.time())
True
>>> my_birthday = date(today.year, 6, 24)
>>> if my_birthday < today:
...     my_birthday = my_birthday.replace(year=today.year + 1)
>>> my_birthday
datetime.date(2008, 6, 24)
>>> time_to_birthday = abs(my_birthday - today)
>>> time_to_birthday.days
202
```

# Öppna filer

- För att kunna använda filer som i/o
- Filen öppnas typiskt med `with open`

# öppna filer

```
f = open('resultat.txt', 'r', encoding='latin_1')
print(f) # skriver ut filobjektet som vi öppnat
print()
content = f.read() # läser innehållet i filobjektet
print(content) # skriver ut innehållet i filobjektet
f.close()
```

```
<_io.TextIOWrapper name='resultat.txt' mode='r' encoding='latin_1'>

Mikaela Andersson       14   0  17
David Fors               8  12  11
Klas Hägglund            0  13   9
Oskar Holmqvist          9  12  10
Gustaf Karlsson         12  14  13
Annica Rundgren         17  10  15
Marcus Carlsson         12   9  12
Johanna Rönnberg         9  15  13
Josefine Carlson        18  14  16
Magnus Ahlsten          13  18  14
Jerker Leo               0   0  11
Jan-Olof Kärrsgård       8  11  13
Siv Tidblom             11   9  12
Mats Karlzon            15  17  10
```

# bättre sätt att öppna fil med `with`

nyckelordet `with` hanterar automatiskt hur länge filen är öppen, även om något går fel.

```
with open ('resultat.txt', 'r', encoding='latin_1') as f:
    print(f)
    print()
    content = f.read()
    print(content)
```

```
<_io.TextIOWrapper name='resultat.txt' mode='r' encoding='latin_1'>

Mikaela Andersson       14   0  17
David Fors               8  12  11
Klas Hägglund            0  13   9
Oskar Holmqvist          9  12  10
Gustaf Karlsson         12  14  13
Annica Rundgren         17  10  15
Marcus Carlsson         12   9  12
Johanna Rönnberg         9  15  13
Josefine Carlson        18  14  16
Magnus Ahlsten          13  18  14
```

# filemodes

- Filer kan öppnas i olika lägen
- läs, skriv etc

| Character | Meaning |
|---|---|
| `'r'` | open for reading (default) |
| `'w'` | open for writing, truncating the file first |
| `'x'` | open for exclusive creation, failing if the file already exists |
| `'a'` | open for writing, appending to the end of the file if it exists |
| `'b'` | binary mode |
| `'t'` | text mode (default) |
| `'+'` | open a disk file for updating (reading and writing) |

# att skriva till en fil

```
# öppna för läsning
with open ('resultat.txt', 'r', encoding='latin_1') as f:
    content = f.read()

#öppna för trunkering och skrivning
with open('new_resultat.txt', 'w') as f:
    print(content, file=f)

#öppna för att skriva till slutet av fil
with open ('new_resultat.txt', 'a') as f:
    print(content, file=f)
```

```
# öppna för ändring
with open ('resultat.txt', 'r', encoding='latin_1') as f:
    content = f.read()

#öppna för trunkering och skrivning
with open('new_resultat.txt', 'w') as f:
    print(content, file=f)
```

# CSV

## Comma Separated Values

```
import csv
with open('fil.csv') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

# CSV

- Typiskt bra för att läsa csv-filer
- Skriva till csv-filer
- Bra om man vill interagera med exempelvis excell
- Mycket lös standard

# Pickle

https://docs.python.org/3/library/pickle.html

- Spara ett objekt
- ’serialize’
- obs byte
- kan spara nästan alla pythons datatyper

# pickle

For the simplest code, use the `dump()` and `load()` functions.

```
import pickle

# An arbitrary collection of objects supported by pickle.
data = {
    'a': [1, 2.0, 3, 4+6j],
    'b': ("character string", b"byte string"),
    'c': {None, True, False}
}

with open('data.pickle', 'wb') as f:
    # Pickle the 'data' dictionary using the highest protocol available.
    pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)
```

The following example reads the resulting pickled data.

```
import pickle

with open('data.pickle', 'rb') as f:
    # The protocol version used is detected automatically, so we do not
    # have to specify it.
    data = pickle.load(f)
```

# json

- problem med pickle:
  - pickle är python specifikt
  - inte säkert att läsa externa filer
  - inte human readable
- json är:
  - human readable
  - säkert
  - går att läsa mellan språk

# ett sätt att koda objekt som text

```
import json
print(json.dumps({'4': 5, '6': 7}, sort_keys=True, indent=4))
```

# dump/dumps

konverterar ett objekt till en text, eller konverterar till text och skriver till en fil

```
#skapa en sträng
import json
gameboard = {"player": "#", "box": "*"}
serialized_dictionary = json.dumps(gameboard, indent=2)
print(serialized_dictionary)
```

```
{
  "player": "#",
  "box": "*"
}
```

```
import json
gameboard = {"player": "#", "box": "*"}
with open('gameboard.txt', 'w') as f:
    json.dump(gameboard, f, indent=2 )
```

# load/loads

konverterar en text i json-format till ett pythonobjekt

```
#loads läser från en sträng
serialzed_dictionary = '{"player": "#", "box": "*"}'
gameboard = json.loads(serialized_dictionary)
print(gameboard)
```

```
{'player': '#', 'box': '*'}
```

```
#load läser från en fil
with open('gameboard.txt', 'r') as f:
    gameboard = json.load(f)
print(gameboard)
```

# Alla funktioner finns i dokumentationen

https://docs.python.org/3/library/json.html

LINKÖPING UNIVERSITY

# Kommandoradsargument

```
python3 test.py argument1 argument2
```

- när ett program startas kan argument skickas med
- viktigt för att skriva program som är användbara
- modulen sys ger oss tillgång till denna och annan data som lagras av interpretatorn

```
import sys
for arg in sys.argv:
    print(arg)
```

```
try.py
argument1
argument2
```

# Exempel, kontrollera att antalet argument stämmer

```
import sys
if len(sys.argv) < 2:
    print("You need to give a filename when running the program")
else:
    with open(sys.argv[1], 'r') as f:
        for line in f:
            print(line)
```

# argparse

- argparse är ett kraftfullt sätt att hantera argument
- svårare att komma igång med, enklare för komplexa fall

Ett program som antingen summerar heltal eller skriver ut det största inmatade talet:

```
import argparse
parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=True, default=False,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.integers)
print(args.accumulate)
```

# argparse

```
import argparse
parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=True, default=False,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.integers) #skriv ut siffrorna som matas in
print(args.accumulate) #skriv ut sant om flaggan --sum skickades med
```

```
python3 try.py 1 2 1 4 1 --sum
[1, 2, 1, 4, 1]
True

python3 try.py 1 2 1 4 1
[1, 2, 1, 4, 1]
False
```

# argparse

```
import argparse
parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=True, default=False,
                    help='sum the integers (default: find the max)')
#...
```

```
python3 try.py 1 2 1 4 1 -h

usage: try.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
  N           an integer for the accumulator

optional arguments:
  -h, --help  show this help message and exit
  --sum       sum the integers (default: find the max)
```

# argparse

- Finns massor av options och sätt att lägga till argument på
- titta i dokumentationen för att hitta det som passar dina behov

https://docs.python.org/3/library/argparse.html

1 Standardbibliotek
2 Moduler
Spara data mellan körningar
3 Kommandoraden
4 Smått och gott
Http
Loggning
5 Undantag

# getpass

- Ibland vill man läsa in lösenord
- Döljer inmatningen

```
import getpass
pwd = getpass.getpass("skriv in lösenord: ")
print(pwd)
```

# subprocess - anrop till operativsystemet

- modul som sköter anrop till operativsystemet
- tar en lista som parametrar i samma stil som sys.argv
- returnerar 0 för lyckad körning
- returnerar 1 för mysslyckad körning

```
subprocess.call(['ls', '-l'])
subprocess.call(['cat', '~/.bashrc]')
```

# Modulen os

- Modulen `os` har operativsystemsspecifika funktioner
- Vissa av funktionerna fungerar endast för specifika operativsystem men de flesta fungerar lika oberoende av operativsystem

http://docs.python.org/3/library/os.html

# Några bra os-funktioner

- Några bra funktioner:

| | |
|---|---|
| os.environ | En dict-liknande typ som innehåller Pythons miljövariabler |
| os.listdir(path) | Utför `ls` i mappen `path`, ger en lista som resultat |
| os.mkdir | Skapar mapp `walk` går sedan nedåt i trädet rekursivt. |

# os.path

- `os.path` är en undermodul med funktioner för att hantera filnamn på ett generellt sätt. Exempel:
  - `os.path.join(p1,p2)` som slår samman sökvägarna p1 och p2 med ett för nuvarande operativsystems bra sätt (med avsende på separator, ’/’ i Linux och ’\’ i windows)
  - `os.path.isfile(f)` kontrollerar om f är en fil
  - `os.path.abspath(path)` gör om `path` till en absolut sökväg

# Webåtkomst

urllib

- Med `urllib` kan man komma åt information från webben
- I modulen `urllib.request` finns en funktion `urlopen` som laddar en sida och ger ett fil-liknande värde som kan hanteras som en textfil

`http://docs.python.org/3/library/urllib.html`

# Webåtkomst

```
import urllib.request
url="http://www.ida.liu.se"
with urllib.request.urlopen(url) as page:
  for line in page:
    if '<title>' in line:
      title = line.split('<title>')
      title = title[1].split('</')
      title = title[0]
      print(line)
      break
```

```
...
<title>IDA > Home</title>
...
```

```
IDA > Home
```

# Loggning

logging

- Med `logging` kan man på ett enkelt sätt logga information om sin körning till fil

```
import logging
logging.basicConfig(filename='example.log',
                    level=logging.INFO)
logging.debug('This does not go to the log file')
logging.info('But this should')
logging.warning('And this, too')
```

```
INFO:root:But this should
WARNING:root:And this, too
```

- Finns stöd för eget format (t.ex. tidsstämplar)

LINKÖPING UNIVERSITY

# Felhantering

- Ibland går inte allt som vi tänkt oss.
- Att programmet kraschar är ok (bra till och med) när vi utvecklar
- Att programmet kraschar är inte alltid ok i verkligheten

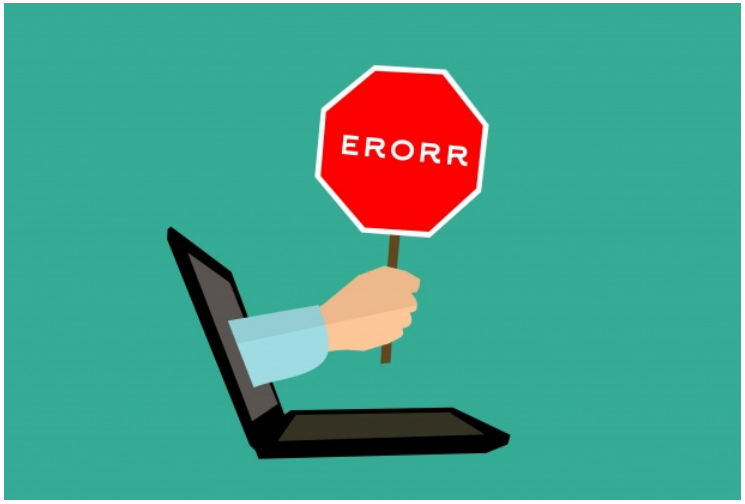
# Try

- inneslut fallet som skall hanteras i ett `try-block`

```
try:
  #koden som kan ge ett fel
  x = int(input('mata in heltal: ')
  print(10/x)
except:
  #här hanterar vi felet
  print('Mata inte in talet 0')
```

# Vad är ett fel?

- Ett objekt
- Kan ge oss information
- har en typ



# Några exempel

## Exception hierarchy

The class hierarchy for built-in exceptions is:

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
```

# Fånga ett fel

```
try:
    x = int(input("Mata in: "))
    print(5/0)
except Exception as e:
    print('något gick fel')
    print(type(e))
```

# Men detta fångar alla fel...

```
try:
    x = int(input("Mata in: "))
    print(5/0)
except Exception as e:
    print('något gick fel')
    print(type(e))
```

# Men vi kan göra det mer specifikt (bättre)

- Tänk på ordningen (mer specifikt först)
  - precis som i if-satser
  - fånga 0-division

```
try:
    x = int(input("Mata in: "))
    print(5/0)
except ZeroDivisionError as e:
    print(type(e))
    print('du får inte mata in 0')
except Exception as e:
    print('något gick fel')
    print(type(e))
```

# Hantera alla fallen

- fånga om användaren inte matar in en siffra

```
try:
    x = int(input("Mata in: "))
    print(5/0)
except ZeroDivisionError as e:
    print(type(e))
    print('du får inte mata in 0')
except ValueError as e:
    print(type(e))
    print('du måste mata in ett tal')
except Exception as e:
    print('något gick fel')
    print(type(e))
```

# Att tänka på

- Använd det inte som en styrstruktur (i princip)
- https://docs.python.org/3/tutori

www.liu.se

LINKÖPING UNIVERSITY