Föreläsning 4

DD1310
Programmering / OPEN
6 hp

Vanligaste metoder

- Viktigaste metoder
  - *li.append(3.14)*
  - *li.extend([123, 'PI'])*
  - *li.insert(2,'Hej hopp')*
  - *print (li.index('tut'))*
  - *li.remove(False)*

Innehåll

- List
- Dictionary
- Felhantering
- Strängjämförelser
- split()
- Filer

Exempel

- orden = ['Liten', 'tuva', 'välter', 'ofta', 'stort', 'lass']
- for ord in orden:
- print (ord)
- for i in range(len(orden)):
- print (orden[i])

List

- Datatypen "list" är en numrerad lista med värden.
  - Varje värde kallas för ett element
  - Värdets position kallas för ett index.
  - En lista med N element har index från 0 upp till N-1.
- En lista är ett objekt och man kan skapa en tom lista utan element genom att skriva
  - *li = list();*
- Sedan kan elementen tilldelas värden:
  - *li.append(3.14)*
  - *li.append('tut')*
  - *li.append(False)*
- Alternativt kan man ge listan värden från början:
  - *li = [3.14, 'tut', False]*
- Använd funktionen *len* för att få fram längden:
  - *for i in range(len(li)):*
  - print (li[i])

Matris

- En lista med listor kallas matris
- Dubbla *for*-slingor krävs vid t ex utskrift
- En matris kan ritas som ett rutnät
- Användbart på många P-uppgifter

Dictionary

- En oornad samling värden ordnade i par
- Består av en nyckel och ett värde
- Nyckeln kan vara (främst) int och str

Exempel

```
# exception.py

ålderText = input('Hur gammal är du? ')
try:
    ålder = int(ålderText)
except ValueError:
    print ('Det här är inget heltal!')

print 1/0
```

Exempel

```
#d = {}
d = {'age':40}
d['namn'] = 'Sten'
d['jobb'] = 'KTH'

print (d)
print (len(d))

for key in d:
    print (d[key])

del d["age"]
```

Strängjämförelser

- För att jämföra strängar kan man skriva
  'Ada' < 'Beda'
  'ada' < 'Beda'
  'Ada'.upper() == 'ADA'

Felhantering

- Ett exekveringsfel i python kallas för ett *exception*.
- Det finns många fördefinierade exception t ex
  - ValueError – t e x om en sträng inte kan tolkas som ett tal vid konvertering
  - ZeroDivisionError - uppstår t ex vid division med 0
- När ett exception uppstår kan man välja mellan att "kasta det" eller att hantera det.
- Kasta
  - Som standard kastas felet till anropande metod om inget skrivs.
  - Om man har flera anropsnivåer sker detta hela vägen upp till huvudprogrammet, om man inte hanterar felet där heller kraschar programmet.
- Hantera
  - Med *try:* inkapslar man den "känsliga" koden.
  - Med except: fångar man det exception som skett samt skriver eventuellt ett felmeddelande, p s s slipper man exekveringsfel.

split()

- Metoden split() i klassen String används för att dela upp en sträng i delar (tokens), t ex ta fram orden ur en mening.
- Tecknet som ska dela strängen skickas med som parameter
- Returtypen är en lista med beståndsdelarna, t e x orden

Exempel

```
# split.py

mening = "Liten tuva välter ofta stort lass!"
orden = mening.split(' ')
for i in range(len(orden)):
    print (orden[i])
```

Filer

- Vill man spara data mellan programkörningar måste man lagra data på fil. Läsning från fil har många likheter med att läsa från tangentbordet och skrivning till fil har många likheter med att skriva till skärmen.
- Man öppnar en fil för läsning / skrivning med funktionen open().
- När man arbetar klart med filen stänger man den för vidare inläsning / utskrift med close()

Exempel

```
# IO.py

infil = open('infil.txt','r')
utfil = open('utfil.txt','w')

rad = infil.readline()
print (rad)
infil.close()

namn = input('Vad heter du? ')
utfil.write(namn)

utfil.close()
```