TDDE54
Python

Föreläsning 3

Emma Enocksson Svensson & Magnus Nielsen

LINKÖPINGS UNIVERSITET

Sammansatta Datatyper

- Datatyper som består av ett antal data i en behållare

- I Ada:n har ni stött på:
  - (Strängar)
  - Fält
  - Poster

I Ada:n har vi använt sammansatta datatyper:

- som har förbestämd storlek
- som har förbestämd datatyp att innehålla
- som har valbara index (eller namn i poster)

I Python däremot…

I Python har sammansatta datatyper…

- ingen förbestämd storlek
- ingen förbestämd datatyp att innehålla
- fasta index

I Python finns (bland mycket annat):

- Listor och Tupler
  - Liknar Ada:s fält
- ”Ordböcker” / Dictionaries
  - Liknar Ada:s poster

Listor

- Annoteras med [ ]
  Ex: my_list = []
- Dynamisk storlek
- Kan innehålla olika datatyper samtidigt
- Är noll-indexerade (likt strängar)
- Det finns många färdiga funktioner som kan hantera listor

Intervallindexering

- Listor kan, likt strängar, intervallindexeras.
- Ex:
  my_list = [”a”, ”b”, ”c”, ”d”, ”e”]
- print(my_list[1:3])
- # vi får utskriften: ['b', 'c']
-
  Intervall, likt range, är ”från och med” och ”fram till”.
  my_list[1:3]
  Vi säger: my_list index 1 fram till 3, ej inklusive 3.
- Kan utelämna ett av indexen:
  my_list[:3] # alla index fram till 3
  my_list[3:] # index 3 och framåt, till slutet
  Kan likt loopar backas, se exempel

Listor

```
list = ”A”, ”B”, ”B”, ”C”]
```
```
["A", "B", "B", "C"]
```

```
list[0] = 8
```
```
[8, "B", "B", "C"]
```

```
list.append(4)
```
```
[8, "B", "B", "C", 4]
```

```
list.remove(”B”)
```
```
[8, "B", "C", 4]
```

```
del list[1:3]
```
```
[8, 4]
```

```
list.sort()
```
```
[4, 8]
```

Indexering

```
list = ["A", "B", "C"]          ["A", "B", "C"]

list[0]                                  "A"
list[-1]                                 "C"

list[0:2]                          ["A", "B"]

list[1:]                           ["B", "C"]
list[:2]                           ["A", "B"]
```

Indexering - steg

```
list = ["A", "B", "C", "D"] ["A", "B", "C", "D"]

list[0:4:2]                          ["A", "C"]

list[3:0:-1]                      ["D", "C", "B"]
list[3::-1]                    ["D", "C", "B", "A"]
```

Summering av intervallindexering

- Vi har 3 möjliga operander i intervallindexering:
  - Startindex
  - Slutindex
  - Steg
- Slutindex -1 ger slutet på listan
- Steget kan vara negativt för att få ut innehållet baklänges
- Vi kan utelämna endera index för att antingen gå från start till givet index, eller från startindex till slutet.

Matriser

```
list = [["A", "B", "C"],  [["A", "B", "C"],
        ["D", "E", "F"],   ["D", "E", "F"],
        ["G", "H", "I"]]   ["G", "H", "I"]]
```

```
list[0][1]                                  "B"
```

Tupler

- Skapas med ( )
- Har bestämd längd
- Ändra data i tuple? **Vi återkommer!**
- Indexeras likt listor
- Kan konkateneras med +

Tupler

```
tup1 = (”A”, [1, 2], ”C”)
tup2 = (”D”, (”E”, ”F”))

print(tup2[1])                    (”E”, ”F”)

tup3 = tup1 + tup2 (”A”, [1, 2], ”C”, ”D”, (”E”,
                                        ”F”))
```

Ordbok / Dictionaries

- Kallas också Maps, Hashmaps eller Dictionaries
- Skapas med { }
- Använder sig av en nyckel som kopplas till värdet
- Dynamisk storlek
- Värdena kan vara olika datatyper
- Nycklarna måste vara unika och samma datatyp

Ordbok

```
dict = {”n1”: 6,
        ”n2”: ”E”}
```

```
{”n1”: 6,
 ”n2”: ”E”}
```

```
dict[”n1”] = ”H”
```

```
{”n1”: ”H”,
 ”n2”: ”E”}
```

```
dict[”n3”] = ”J”
```

```
{”n1”: ”H”,
 ”n2”: ”E”,
 ”n3”: ”J”}
```

Ordbok

```
dict = {”n1”: 6,
        ”n2”: ”E”}
```

```
{”n1”: 6,
 ”n2”: ”E”}
```

```
dict.keys()
```

```
[”n1”, ”n2”]
```

```
dict.values()
```

```
[6, ”E”]
```

```
del dict[”n1”]
```

```
{”n2”: ”E”}
```

Gemensamma funktioner

- len
  - Returnerar längden på en sammansatt datatyp
- Iteratorer
  - Hjälper till för att kunna loopa över sammansatta datatyper

Iteration över sammansatta datatyper

Iteratorer kan användas för att loopa över sammansatta datatyper, dock ej för att modifera enskilda data.

```
list = [”A”, ”B”, ”C”]
for i in range(len(list)):
   print(list[i])
```

Vs.

```
for item in list:
   print(item)
```

Bonus: Strängar

- Strängar är i grunden listor
- Allt ni kan göra med indexering av listor kan ni göra med en sträng
- len fungerar för strängar
- Det finns skillnader!

Sammansatta datatyper som parametrar

- Modifiering av innehåll:
  - Modifiering m.h.a. = -> skapar istället en lokal variabel med samma namn som parametern
  - Modifiering m.h.a. indexering -> Ändrar värdet i parametern. Påminner om Ada:s out
  - Modifiering m.h.a. append eller dylikt -> Ändrar innehållet i parametern. Påminner också om Ada:s out

Tupler

```
tup1 = (”A”, [1, 2], ”C”)
```

```
tup1[1].append(3)      (”A”, [1, 2, 3], ”C”)
```

Dagens Uppgift

Låt oss skapa en swap-funktion

Extrauppgift

Skapa en datastruktur som ska representera en student i kursen TDDE54.
Skapa passande funktioner.

emma.enocksson@liu.se
magnus.nielsen@liu.se

www.liu.se

LINKÖPINGS UNIVERSITET