Övning 1 - Abstrakta datatyper

```
0 # coding: latin
```

Summering

Vi gick igenom betydelsen av **abstrakta datatyper/datastrukturer**. Detta exemplifierades genom att dels titta på generella abstrakta dataytyper så som *stacken*, *inläsning och utskrift* och *listan* samt dels studerade en egendefinerade abstrakta datatyper *fraction*. I pdfen finns två ytterligre exemple på egendefinerade abstrakta datayper *temperatur* och *iso2utf*.

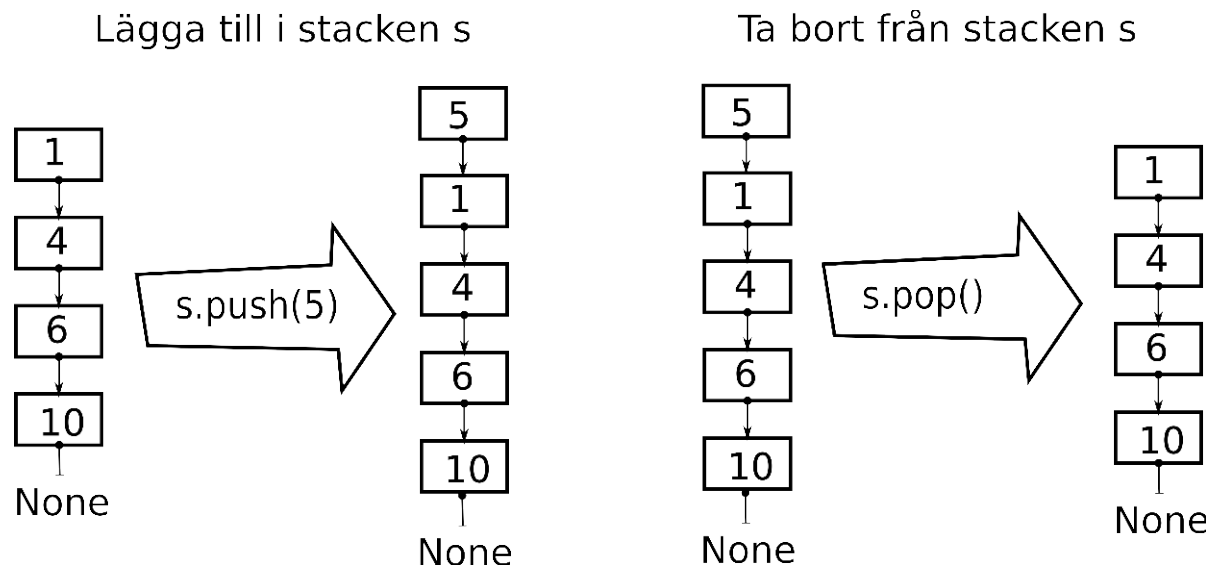
```
18
19 import sys, os
20 sys.path.append(os.getcwd())
21 from show_fig import show_fig
```

1 Stacken

- Rita bilder som visar hur det ser ut när man lägger in ett nytt element i stacken.
- Rita bilder som visar hur det ser ut när man tar bort ett element från stacken.
- Skriv kommentarer till koden nedan!
- Skriv ett program som använder två stackar för att lösa något problem.

In och uttag från stacken

```
33
34 show_fig('stack_example.png')
```




```
34
35 # -*- coding: utf-8 -*-
36 """Classroom exercise 1, example 1."""
37
```

```
38 class Stack:
39     """A representation of a last-in-first-out (LIFO) stack of
40     objects."""
41     def __init__(self):
42         """Create an empty stack."""
43         self.top = None
44 
45     def push(self,x):
46         """Push an item onto the top of this stack."""
47         ny = Node(x)
48         ny.next = self.top
49         self.top = ny
50 
51     def pop(self):
52         """Remove the top object from this stack and return it."""
53         x = self.top.value
54         self.top = self.top.next
55         return x
56     def isempty(self):
57         """Test if this stack is empty."""
58         ## The naïve solution (below) is unnecessary:
59         # if self.top == None:
60         #     return True
61         # else:
62         #     return False
63         return self.top == None
64 
65 class Node:
66     """An entry in a linked structure (e.g. Stack)."""
67     def __init__(self, x):
68         """Create a node linked to the given element."""
69         self.value = x
70         self.next = None
```

Med help kan vi nu skriva ut en kortfattad beskrivning av programmet

```
73 print(help(Stack))
```

```
Help on class Stack in module pyreport.main:

class Stack
 |  A representation of a last-in-first-out (LIFO) stack of
 |  objects.
 |
 |  Methods defined here:
 |
 |  __init__(self)
 |      Create an empty stack.
 |
 |  isempty(self)
 |      Test if this stack is empty.
 |
 |  pop(self)
 |      Remove the top object from this stack and return it.
 |
 |  push(self, x)
 |      Push an item onto the top of this stack.

None
```

Exempel användning av stacken

```
79
80 data = [1,2,3,4]
81 s1 = Stack()
82 for number in data:
83     print "pushing on s1:", number
84     s1.push(number)
```

```
pushing on s1: 1
pushing on s1: 2
pushing on s1: 3
pushing on s1: 4
```

```
85
86 s2 = Stack()
87 while not s1.isempty():
88     number = s1.pop()
89     print "pushing", number, " on stack s2"
90     s2.push(number)
```

```
pushing 4  on stack s2
pushing 3  on stack s2
pushing 2  on stack s2
pushing 1  on stack s2
```

```
91
92 print "s1 isempty?", s1.isempty()
```

```
s1 isempty? True
```

```
92 print "s2 isempty?", s1.isempty()
```

```
s2 isempty? True
```

2. Inläsning och utskrift

```
97
98 raden = raw_input("Ge en rad tal: ")
```

```
Ge en rad tal:
```

```
98 raden_i_delar = raden.split()
99 for tal in raden_i_delar:
100     print tal
```

```
1
34
5
6
7
```

Listan - exempel på en datastruktur i Python

Data: ett antal tal, tecken eller objekt
Metoder och operationer:

- append(x) Lägger till x sist i listan.
- insert(i,x) Lägger till x på plats i.
- pop() Plockar bort sista elementet från listan.
- sort() Sorterar listan.

- reverse() Vänder på listan.
- index(x) Returnerar index för första förekomsten av x.
- count(x) Räknar antalet x i listan.
- remove(x) Tar bort första förekomsten av x.
- [i] Returnerar i:te elementet i listan.
- [:] Plockar ut en dellista.
- – Konkatenerar två listor
- – Flerdubblar listan.
- in Kollar om ett element finns med i listan.
- len Returnerar listans längd.

3. Läs från fil och sortera

```
127 infil = open("bilar.txt")
128
129 lista = []
130 for rad in infil:
131     ordet = rad.strip()
132     lista.append(ordet)
133 lista.sort()
134
135 print lista
```

```
['Bmw', 'Fiat', 'Porsche', 'Saab', 'Volvo']
```

4. Jämföra inläsning från text mot inläsning från web

```
139 import urllib
140 import time
141
142 def lasFilSortera(filnamn):
143     infil = open(filnamn)
144     lista = []
145     for rad in infil:
146         ordet = rad.strip()
147         lista.append(ordet)
148     lista.sort()
149     return lista
150
151 def lasURLSortera(url):
152     infil = urllib.urlopen(url)
153     lista = []
154     for rad in infil:
155         ordet = rad.strip()
156         lista.append(ordet)
157     lista.sort()
158     return lista
159
160
161 innan = time.time()
162 lista1 = lasFilSortera("tred.txt")
163 efter = time.time()
164 print "Tid för filläsning:", efter - innan
```

```
Tid för filläsning: 0.000543117523193
```

```
166
167 innan = time.time()
168 lista2 = lasURLSortera("http://www.csc.kth.se/utbildning/kth/kurser/DD1320/
    til\
169 a11/ovn/tred.txt"                )
170 efter = time.time()
171 print "Tid för webbläsning:", efter - innan
```

```
Tid för webbläsning: 0.0210530757904
```

Egna datastrukturer

I kursen ska ni själva implementera datastrukturer, oftast genom att skriva egna klasser i Python.
Abstrakt datatyp för temperatur

- Temperatur kan anges i olika skalor. En abstrakt datatyp minskar risken för missförstånd. Definiera klassen temp med metoderna setK, setC, setF och getK, getC, getF.
- Använd sedan klassen i ett program som läser in utomhustemperaturen (Celsius) och skriver ut temperaturen så att en amerikan förstår (Fahrenheit).

Filen temp.py:

```
192
193 """Temp - an abstract datatype for temperature
194 Temp is used to represent temperature in multiple scales.
195 """
196
197 ZERO_C = 273.15  # 0 C == 273.15 K
198
199 ZERO_F = 459.67*5/9  #0 F == 255.37222... K
200
201
202 # Conversion between Kelvin, degrees Celsius, degrees Fahrenheit
203 # [C] = [K] + 273.15            [K] = [C] - 273.15
204 # [C] = ([F] - 32)*5/9          [F] = [C]*9/5 + 32
205 # [K] = ([F] + 459.67)*5/9      [F] = [K]*9/5 - 459.67
206
207 class Temp:
208
209     def __init__(self):
210         self.K = 0                      #Temperatur i Kelvin
211
212
213     def setK(self,K):
214         self.K = K
215
216     def setC(self,C):
217         self.K = ZERO_C+C
218
219     def setF(self,F):
220         self.K = ZERO_F+5*F/9
221
222     def getK(self):
223         return self.K
224
225     def getC(self):
226         return self.K-ZERO_C
227
228     def getF(self):
229         return (self.K-ZERO_F)*9/5
```

Import temp

```
226
227 from temp import Temp
228 t = Temp()
229 c = input("Vad är temperaturen i Celsius? ")
```

```
Vad är temperaturen i Celsius?
```

```
229 t.setC(c)
230 print "Amerikaner kallar det", str(t.getF()) + "F"
```

```
Amerikaner kallar det 86.0F
```

```
230 # Vad är temperaturen i Celsius? 1
```

6. Fraction-klassen: Ett exempel från Miller & Ranums bok, s 35:

```
236     # encoding: Latin1 (tillåt å, ä och ö)
237     # Fraction-klassen
238
239     class Fraction:
240
241         def __init__(self, top, bottom):
242             self.num = top
243             self.den = bottom
244
245         def __str__(self):
246             return str(self.num)+"/"+str(self.den)
247
248         def show(self):
249             print self.num,"/",self.den
250
251         def __add__(self,otherfraction):
252             newnum = self.num*otherfraction.den +                    self.den*
                    otherfraction.num
253             newden = self.den * otherfraction.den
254             return Fraction(newnum,newden)
255
256         def __cmp__(self,otherfraction):
257             num1 = self.num*otherfraction.den
258             num2 = self.den*otherfraction.num
259             if num1 < num2:
260                 return -1
261             else:
262                 if num1 == num2:
263                     return 0
264                 else:
265                     return 1
```

Svara på följande frågor:

1. Vad är relationen mellan en klass och ett objekt? - objekt är instantieringar av klasser, engelskan "instance" . Hur skriver man för att skapa ett Fraction-objekt? - fr=Fraction(2,3) . Vad gör $_{init_{?-anropasnrettobjectskapas,stterattributen}}$

7. Text konvertering

Följande funktion konverterar en sträng från teckenkodningen “iso8859-1” till “utf-8”. Gör funktionen generellare så att den kan konvertera från en valfri given teckenkodning till en annan.

```
302
303 def iso2utf(strang):
```

```
304     """
305     Konverterar från iso8859 till utf-8
306     """
307     ustrang = strang.decode('iso8859-1')
308     return ustrang.encode('utf-8')
```

Konvertering av valfri teckenkodning

```
316 def convert_caracter_set(strang, current_set, target_set):
317     """
318      Convert from one caracter set to another
319 
320      Input:
321          string - string to convert
322          current_set - current caracter set as a string, e.g. iso8859
323          target_set - target caracter set as a string, e.g. utf-8
324 
325     """
326     ustrang = strang.decode(current_set)
327     return ustrang.encode(target_set)
```