```
0 # coding:latin
```

Övning 6

Idag delades P-uppgifteran ut samt vi hade frågestund. Vidare gick vi igenom Schampometoden, namngivning, kommentarer och pydoc.

P-del (LAB3)

Kursens tredje moment, LAB3, är en större, individuell programmeringsuppgift i Python; en "P-uppgift". P-delen redovisas i tre steg under kursens andra period. Uppgifterna är tänkta att vara något så när svåra och tidskrävande, räkna med ca 80 timmar för en medelsvår uppgift.

- Specifikation: Innan programmet skrivs ska en specifikation redovisas. Syftet med specifikationen är att du ska tänka igenom problemet innan du försöker lösa det.
- Granskning: Innan det färdiga programmet kan redovisas för en handledare ska det granskas av en annan kursdeltagare som tittar på koden och provkör programmet. Det är obligatoriskt för varje kursdeltagare att granska ett program.
- Slutredovisning: Du väljer (normalt via webben) en tid för slutredovisning. Uppgiftslydelsen, specifikationen, besiktningsprotokollet och granskaren ska medföras till slutredovisningen, liksom en färsk programutskrift, källkod och det körbara programmet. Om du tar med en egen dator till redovisningen ska den vara uppkopplad mot Internet, eftersom vi jämför med tidigare lösningar (för att stävja fusk).

P-uppgifterna är av olika omfång och kräver olika mycket tid.

Som ett komplement till de "inbyggda" finesserna i Python så finns ytterligare Pythonfiler för P-uppgiften. Utnyttja gärna dessa, men tänk på att i enlighet med hederskodexen ALLTID ange varifrån koden kommer när det inte är din egen.

Efter kursens slut kan P-delen endast redovisas i omtentaperioder och ger maximalt betyget D. Weblaborationer (LAB4) En fristående modul för MEDIA. Fyra av kursens föreläsningar tillägnas genomgång av de verktyg som krävs för att lösa de två laborationer denna består av. Med fristående menas här att denna del endast i ringa grad bygger på kunskap från övriga kursmoment.

Hederskodex

Skolan tillämpar en hederskodex i alla sina kurser och varje student förutsätts tillämpa hederskodexen. Den finns via länk på kursens webbsida.

Betyg

Betyget på kursen avgörs helt av P-uppgiften. De flesta uppgifterna kan byggas på för att ge högre betyg. I övrigt gäller följande:

> E = Godkänd P-redovisning av uppgift som lösts tillräckligt bra (max tre påpekanden).
>
> D = Godkänd P-redovisning med ett perfekt program, dvs inga anmärkningar i protokollet (väl uppdelat, ingen koduppprepning, vettigt dokumenterat mm).
>
> För betyg högre än D krävs att grunduppgiften redovisas före kursomgångens slut. Kraven på ett perfekt program gäller hela programmet, inklusive extrauppgifter.
>
> C = Kraven för D + en extrauppgift med betyg C (ofta hantering av felaktig inmatning).
>
> B = Kraven för C + en extrauppgift med betyg B (ofta en svårare algoritm).
>
> A = Kraven för B + en extrauppgift med betyg A (ofta grafiskt användargränssnitt eller avancerad algoritm).

Om du är missnöjd med övningsassistentens beslut om betyg kan du skicka ett e-brev till kursledaren där du förklarar din syn på betyget och bifogar alla filer.

Plussa

På KTH finns av hävd en rätt att tenta om för att få högre betyg fast tentan redan är godkänd. I kurser med andra typer av examination (än tenta) måste vi av kostnadsskäl begränsa denna rätt. Följande regler gäller i denna kurs:

- Den som har redovisat sin P-uppgift under kursens gång kan efter det höja sitt betyg, men bara inom ett år från kursstart.
- Krav för betygen framgår av kursprogrammet. Man kan behöva förbättra programkvalitén eller göra vissa extrauppgifter. Man kan också göra en helt ny uppgift.
- Redovisningen görs vid något av uppsamlingstillfällena för P-uppgifter.
- Man har bara ett år från kursstart på sig att höja sitt betyg, sen är det försent.

Vanliga frågor:

Får de byta P-uppgift senare? *Ja, då får de prata med Sten om det, fast det kanske inte är så många lydelser kvar*

Får de hitta på en egen P-uppgift *Ja, då får de prata med Sten om det*

Får de samarbeta med en kompis som löser samma P-uppgift *Nej, och de ska inte ens sitta bredvid varandra - P-uppgiften är en personlig uppgift*

Ska jag hålla reda på vem som valde vilken uppgift? *Nej*

Programutveckling med SCHAMPOMETODEN

Programutveckling är en konst som kräver mycket mer än att man behärskar sitt programspråk. Vägen från problem till program är kantad av otaliga frågor. Hur ska man veta vilka procedurer och funktioner som man bör definiera? I vilken ordning ska man skriva programmet? Ska man tänka och rita först eller ska man skriva pythonkod direkt? Som grön programmerare önskar man att det funnes någon allmän metod att hålla sej till. Det gör det den heter SCHAMPO!

Det är ett minnesord med fem ljud som ska påminna om dom fem stegen.

1. Skärmen Rita hur den ser ut efter en lyckad körning. Dä ser man all in- och utmatning som gjorts.
2. Algoritmen Skriv hur man skulle lösa problemet utan dator. I enkla fall är algoritmen en formel.
3. Minnet Skriv upp variabelnamn och vilka värden dom tildelas i ditt körexempel.
4. Procedurer Varje deluppgift får en egen procedur eller funktion. Se dom som medarbetare, specialiserade på var sin uppgift.
5. Oppifrån Skriv nu programsatserna oppifrån och ner.

Namngivning

Stora eller små bokstäver, namngivning, här får ni svaret. Kom ihåg att variabelnamn ska vara informativa, dvs genom att läsa dem ska man få en uppfattning om vad de lagrar. OBS undvik att blanda engelska och svenska. Välj ett av språken:

Classes: CapWords

Exceptions: CapWords

Methods/Functions: lower_with_under()

Global/Class Constants: CAPS_WITH_UNDER

Global/Class Variables: lower_with_under

Instance Variables: lower_with_under

Function/Method Parameters: lower_with_under

Local Variables: lower_with_under

Läs mer på http://google-styleguide.googlecode.com/svn/trunk/pyguide.html

Kommentering

Any function or method which is not both obvious and very short needs a doc string

Problem

ICA vill att det ska vara lätt att sätta sig in i hur deras kund och produkthanteringssytem fungerar. De vill att klasser och funktioner ska vara väl kommenterade och sen generera hjälp dokument utifrån kommentarerna. Till sin hjälp har de modulen pydoc.

```
198 '''
199 Exempel på bra kommentering av klass och funktion.
200
201 Vilka metoder/funktioner ska dokumenteras? All metoder som inte både är både
202 uppenbara och mycket korta.
203
204 Dokumentera alltid klasser
205 '''
206
207 class Affar:
208     """ Affar(affar)
209     Klass som definierar en affär
210
211     Lång beskrivning om det behövs...
212
213     Attributes:
214         affar    - namn på affär
215
216     Example:
217         >>> a1 = Affar('ICA Nortälje')
218     """
219     def __init__(self, affar):
220         self.affar = affar
221
222 class Produkt(Affar):
223     """ Produkt(affar, beskrivning, nr, pris)
224     Klass som definierar en produkt med beskrivning och pris
225
226     Lång beskrivning om det behövs...
227
228     Attributes:
229         affar          - namn på affär
230         beskrivning    - produktbeskrivning
231         nr             - produktnummer
232         pris           - pris på produkten
233
234     Example:
235         >>> p1 = Produkt('ICA Nortälje', 'En kaffekokare', 0, 199.5)
236     """
237     def __init__(self, affar, beskrivning, nr, pris):
238         Affar.__init__(self, affar)
239         self.__beskrivning = beskrivning
240         self.__nr   = nr
241         self.__pris = float(pris)
242
243     def beskrivning(self):
244         return self.__beskrivning
245
246     def nr(self):
247         return self.__nr
248
```

```
249     def pris(self):
250         return self.__pris
251 
252     def __lt__(self, other):
253         return self.__beskrivning < other.__beskrivning
254 
255     def __str__(self):
256         return ('Beskrivning: ' + self.__beskrivning + '\n'
257                 'Produktnummer: ' + str(self.__nr) + '\n'
258                 'Pris: ' + str(self.__pris)+'\n'
259                 'Affar: '+ str(self.affar)+'\n')
260 
261 
262 def lasIn(fileName):
263     """Filinläsning
264 
265     För varje rad i filen splittas raden på / och dellista med
266     strängarna läggs i en lista.
267 
268     Arguments:
269         fileName: namn på fil att läsa in
270 
271     Returns:
272         data: Lista med dellistor som inehåller inlästa strängar. Exempel:
273               [['stol','bord'],['kul','tråkigt']] där ['stol','bord'] och
274               ['kul','tråkigt'] är dellistor.
275 
276     Example:
277         >>> f=open('test','w')
278         >>> f.write('stol/bord\\nkul/tråkigt')
279         >>> f.close()
280         >>> data=lasIn('test')
281         >>> print(data)
282         >>> print(data)
283             [['stol', 'bord'], ['kul', 'tråkigt']]
284     """
285 
286     infil = open(fileName,'r')
287     rad = infil.readline()
288     data = list()
289     while rad != '':
290         rad = rad.rstrip('\n')
291         delar = rad.split('/')
292         data.append(delar)
293         rad = infil.readline()
294     return data
```

Pydoc

```
293 import pydoc
294 
295 print(pydoc.help(Produkt))
```

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

class Produkt(Affar)
 |  Produkt(affar, beskrivning, nr, pris)
 |  Klass som definierar en produkt med beskrivning och pris
 |
```

```
    |  Lång beskrivning om det behövs...
    |
    |  Attributes:
    |      affar           - namn på affär
    |      beskrivning     - produktbeskrivning
    |      nr              - produktnummer
    |      pris            - pris på produkten
    |
    |  Example:
    |      >>> p1 = Produkt('ICA Nortälje', 'En kaffekokare', 0, 199.5)
    |
    |  Methods defined here:
    |
    |  __init__(self, affar, beskrivning, nr, pris)
    |
    |  __lt__(self, other)
    |
    |  __str__(self)
    |
    |  beskrivning(self)
    |
    |  nr(self)
    |
    |  pris(self)
```

Error:

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.6/dist-packages/pyreport-0.3.4c-py2.6.egg/pyreport/main.py", line 180, i
    exec block_text in self.namespace
  File "<string>", line 4, in <module>
  File "/usr/lib/python2.6/pydoc.py", line 1726, in __call__
    self.help(request)
  File "/usr/lib/python2.6/pydoc.py", line 1774, in help
    self.output.write('\n')
ValueError: I/O operation on closed file
```

296
```
print(pydoc.help(lasIn))
```

```
Help on function lasIn in module pyreport.main:

lasIn(fileName)
    Filinläsning

    För varje rad i filen splittas raden på / och dellista med
    strängarna läggs i en lista.

    Arguments:
        fileName: namn på fil att läsa in

    Returns:
        data: Lista med dellistor som inehåller inlästa strängar. Exempel:
              [['stol','bord'],['kul','tråkigt']] där ['stol','bord'] och
              ['kul','tråkigt'] är dellistor.

    Example:
        >>> f=open('test','w')
        >>> f.write('stol/bord\nkul/tråkigt')
        >>> f.close()
        >>> data=lasIn('test')
        >>> print(data)
```

```
>>> print(data)
    [['stol', 'bord'], ['kul', 'tråkigt']]
```

Error:

```
Traceback (most recent call last):
  File "/usr/local/lib/python2.6/dist-packages/pyreport-0.3.4c-py2.6.egg/pyreport/main.py", line 180, i
    exec block_text in self.namespace
  File "<string>", line 3, in <module>
  File "/usr/lib/python2.6/pydoc.py", line 1726, in __call__
    self.help(request)
  File "/usr/lib/python2.6/pydoc.py", line 1774, in help
    self.output.write('\n')
ValueError: I/O operation on closed file
```

Generera html från terminalen för filen exercise.py: 1. user@sdator:~/pydoc exercise6 > exercise6.html 2. user@dator:~/gedit exercise6.html