TDP002 Imperativ programmering

Laborationsmaterial emacs python-mode

Höstterminen 2008

Innehållsförteckning

Introduktion

Detta material innehåller laborationer och övningar för delmomentet emacs python-mode i kursen *TDP002 Imperativ programmering*.

Redovisning av laborationer

Ni ska visa för er labassistent att ni behärskar några av handgreppen i emacs python-mode. Se slutet av denna laborationsanvisning.

Laboration 1 emacs python-mode

Syftet med den här laborationen är att ni ska lära er emacs och pythonmoden i emacs så pass att ni ska kunna arbeta bekvämt med emacs och utan större hinder ska kunna programmera python i emacs.

Uppgift 1 - .emacs

Lägg in i din .emacs så att den stänger av ljudliga bell och slår på matchning av parenteser, paren-mode. Starta om emacs och testa att det fungerar.

Uppgift 2 – Kompilera och kontrollera pythonprogrammet

Spara följande program som vehicle.py. Objektorientering och klasser får ni lära er mer om i andra kurser. En klass kan ses som en komplex datatyp.

```
# -*- coding: iso-8859-1 -*-

class Vehicle:

    def __init__( self, wheels ):
        self.__wheels = wheels  # wheels är ett privat attribut

    def getNumWheels( self ):
        return self.__wheels

class Truck( Vehicle ):

    def __init__( self, doors ):
        Vehicle.__init__( self, 6 )
        self.__doors = doors
        self.__radio = []

    def print_info( self ):
        print "En lastbil har %s hjul.\nDen här har %s dörrar" %
( self.getNumWheels(), self.__doors )

    def print_channels(self):
        for ch in self.__radio:
            print ch

    def add_radio(self):
        self.__radio = ["88", "89","90","P2", "91.1","95.5", "99",
"107.1"]
```

```
class Car( Vehicle ):

    def __init__( self, doors ):
        Vehicle.__init__( self, 4 )
        self.__doors = doors

    def print( self ):
        print "En bil har %s hjul.\nDen här har %s dörrar" %
( self.getNumWheels(), self.__doors )
```

Kontrollera och kompilera pythonprogrammet. Tänk på att print är ett reserverat ord i python. Korrigera de fel ni får och gör om proceduren.

Uppgift 3 – Information om pythonsymboler

Ställ er på en print och plocka fram information om pythonsymbolen.

Uppgift 4 - Indentering

Ändra funktionen print_channels så att den endast skriver ut radiokanalerna om lastbilen har fler än 2 dörrar. Använd pythonmodens möjlighet att ändra indentering (shifta regionen åt höger)

Uppgift 5 – Markera funtioner och klasser

Använd kortkommandot för att markera en funktion och markera funktionen add_radio. Klipp ut funktionen och klistra in den före print_channels.

Använd kortkommandot för att markera en klass och markera och klipp ut klassen Car och klistra in den före klassen Truck.

Uppgift 6 - Pythoninterpretatorn

I denna uppgift ska ni köra pythoninterpretatorn från emacs och ni ska kunna gå bakåt och framåt i listan över tidigare utförda kommandon i denna interpretator och köra dem igen.

En klass kan betraktas som en komplex datatyp. Om man vill skapa ett objekt och tilldela en variabel detta objekt så görs det med:

```
variabeln = klassen()
```

Om ni t.ex. har filen modul.py som innehåller klassen foo och ni har gjort import modul blir det:

```
bar = modul.foo()
```

En klass kan ha metoder som arbetar på datat i den. En metod kan t.ex. vara att ange ålder på en person, skriva ut antalet dörrar som bilen har etc.. Om klassen foo ovan har en metod gazonk går det att för variabeln bar köra funktionen gazonk:

```
bar.gazonk()
```

Starta en pythoninterpretator från emacs.

Importera vehicle

Tilldela variabeln lastbil ett lastbilsobjekt med två dörrar.

```
lastbil = vehicle.Truck(2)
```

För lastbil kör metoden print_info.

```
Lastbil.print_info()
```

För lastbil kör metoden print_channels.

```
lastbil.print_channels()
```

För lastbil kör metoden add_radio.

```
lastbil.add_radio()
```

Kör åter igen metoden print_channels. Skriver den ut kanallistan?

Tilldela variabeln lastbil ett lastbilsobjekt med tre dörrar.

Kör metoden add_radio.

Kör metoden print_channels. Skriver den ut kanallistan?

Uppgift 7 – Information om symboler och komplettering

Lägg till *import re* i början av filen vehicle.py.

I en funktion lägg till re. och använd komplettering av symbol för att få fram vilka alternativ som finns till re. Ställ er sedan på re och plocka fram information om re på samma sätt som ni gjorde tidigare med print.

Uppgift 8 – Extra (frivilliga övningar)

Start en extra frame och ställ er i början av bufferten vehicle.py i denna nya fram. Se hur ni kan se början av en buffert i en frame och t.ex. slutet av samma buffert i en annan frame.

Stäng er nya frame.

Se till att ha ett fönster med python-koden och ett fönster med interpretatorn. Byt fönster med kortkommandon.

Om ni inte redan har gjort det kör igenom emacs tutorial.

Prova och starta tetris med M-x tetris

Prova och prata med psykologen med M-x doctor

Redovisning

Visa för er laborationsassisten att ni behärskar följande:

- Att ni från emacs kan kompilera och kontrollera pythonkoden.
- Att ni kan markera en funktion eller klass och flytta den.
- Att ni kan få fram information om en pythonsymbol.
- Att ni kan ändra indentering på ett avsnitt/block.
- Att ni kan starta interpretatorn och i den kan utföra pythonkommandon och kan repetera

tidigare utförda kommandon.