1. Zoznamy a slovníky

(1) Napíšte funkciu, ktorá dostane ako parameter zoznam a vráti zoznam, v ktorom zlúči rovnaké po sebe idúce prvky.
(2) Napíšte funkciu, ktorá dostane ako parameter dva utriedené zoznamy a vráti zoznam, ktorý tieto dva zoznamy zlúči do utriedeného zoznamu.
(3) Napíšte funkciu `permute(`l`)`, ktorá dostane zoznam l ako parameter a vráti zoznam všetkých permutácií zoznamu l.

Funkcia bude zrejme rekurzívna. Ja som to spravil takto: ak l je prázdny, vráti zoznam obsahujúci prázdny zoznam. Ak l je neprázdny, odoberie z neho prvý prvok a zavolá `permute` *pre zoznam bez prvého prvku. Potom treba tento zoznam prejsť a vrátiť nový zoznam. Dávajte si pozor na to, aby ste na patričnom mieste vytvárali kópiu permutácie $n-1$ prvkov cez operátor* [:]. *Vsúvanie na pozíciu* `poz` *do zoznamu* `a` *sa robí takto:*

```
a[poz:poz]=[prvok]
```

(4) Napíšte funkciu, `dict_join(`d_1`,`d_2`)` ktorá dostane ako parameter dva slovníky d_1, d_2 a vráti slovník d taký, že
   (a) d obsahuje práve tie kľúče, ktoré sú súčasne v d_1 aj d_2,
   (b) $d[k] = (d_1[k], d_2[k])$ pre k `in` d – t.j. hodnota $d[k]$ je n-tica hodnôt.
(5) Rozšírte predošlé cvičenie tak, aby fungovalo pre ľubovoľný počet parametrov.

Viď časť 4.7 Python tutorial, alebo prednáška.

2. Symbolické výpočty

Nasledujúce cvičenia reprezentujú racionálne [1] funkcie prefixovo takto (príklad)

$$\frac{x+2y}{3+\frac{2.5}{x}}$$

je reprezentovaná ako

```
['/',['+','x',['*',2,'y']],['+',3,['/',2.5,'x']]].
```

Tomuto budeme hovoriť prefixová reprezentácia. Uvedomte si, že v tejto reprezentácii je konštantná funkcia všade rovná 5.2 reprezentovaná ako číslo `5.2` a funkcia všade rovná x reprezentovaná ako reťazec `'x'`.

Pre riešenie potrebujete použiť modul `types`. Viď dokumentácia.

Zrejme spracovanie takýchto prefixových reprezentácií bude rekurzívne.

(1) Napíšte funkciu `myevald(f,d)`, ktorá dostane ako parameter funkciu `f` v prefixovej reprezentácii a slovník `d`, ktorý zobrazuje všetky premenné použité v `f` na čísla. Ako návratovú hodnotu vráti hodnotu funkcie `f`.

```
>>> myevald(['+','x','y'],{'x':10,'y':20})
30
```

(2) Napíšte funkciu `myeval` tak, aby fungovala cez pomenované parametre: `myeval(['+','x','y'],x=2,y=5)` vráti 7. Využite predošlé cvičenie.
(3) Napíšte funkciu `compose(f,d)`, ktorá dostane ako parameter funkciu `f` v prefixovej reprezentácii a slovník `d`, ktorý zobrazuje premenné použité v `f` na funkcie v prefixovej reprezentácii. Ako návratovú hodnotu vráti zloženú funkciu v prefixovej reprezentácii. Dajte si pozor, aby ste skladanie robili správne: t.j. dosadzovaním „naraz" a nie „postupne":

[1] Nemusíte implementovať umocňovanie na konštantu, stačí `['+','-','*','/']`.

```
compose(['+','x','y'],{'x':['+','x','y'],'y':'z'})
```

má vrátiť

```
['+',['+','x','y'],'z']
```

a nie

```
['+',['+','x','z'],'z'].
```

Keď to všetko zvládnete, pozrite sa na `http://code.google.com/p/sympy/` .

3. Objekty a triedy

(1) Teraz ideme zabaliť prefixovo reprezentované funkcie tak, aby fungovali „objektovo".

Definujte triedu `class SymbFunc(object)` v module `symb.py` tak, aby fungovalo toto:

```
>>> import symb
>>> f=symb.SymbFunc(['+','x','y'])
>>> f.eval(x=10,y=20)
30
>>> f.evald({'x':10,'y':20})
30
```

(2) Premenujte teraz metódu `SymbFunc.eval` na `SymbFunc.__call__`. Dôsledok je, že to začne fungovať takto:

```
>>> import symb
>>> f=symb.SymbFunc(['+','x','y'])
>>> f(x=10,y=20)
30
```

Viď Python Reference Manual, časť 3.4.4.

(3) Pridajte metódu `__eq__(self,other)`, ktorá vráti `True` ak sú funkcie rovné. Pri určovaní rovnosti funkcií berte do úvahy komutativitu operácií + a *, t.j.

```
>>> f=symb.SymbFunc(['*',['+','x','y'],'z'])
>>> g=symb.SymbFunc(['*','z',['+','y','x']])
>>> f==g
True
```

Ignorujte všetko ostatné okrem komutativity.

(4) Skúste to teraz prerobiť tak, aby fungovalo aj volanie `f(10,20)`. Ktorá hodnota zodpovedá ktorej premennej je dané ich abecedným poradím.

Vhodné je vytvoriť abecedne utriedený zoznam premenných už v `SymbFunc.__init__`.