Skúšobné zadanie na druhé cvičenie

18. mája 2012

1 Príprava

Vaše riešenie bude obsahovať iba definície funkcií. Funkcie sa majú volať tak, ako je zadané. Nemajú nič vypisovať. Nechajte v ňom aj funkciu `myeval` z domácej prípravy (ak ju máte). Neodporúčam postupovať spôsobom „skopírujem `myeval` a začnem upravovať".

Zrejme budete potrebovať vaše riešenie skúšať. V Pythone sa to robí tak, že na koniec vášho modulu dáte konštrukciu tohto typu

```
if __name__=="__main__":
    print testovaci_vypis_1
    print testovaci_vypis_2
    print testovaci_vypis_3
```

Tým dosiahnete, že váš modul bude po spustení na príkazovom riadku vypisovať testovacie výpisy. Zároveň keď ho ja budem importovať v testoch, nebudú mi tam strašiť vaše testovacie výpisy.

2 Substituujte (6 bodov)

Napíšte funkciu, `compose(f,d)`, ktorá dostane funkciu `f` v prefixovej reprezentáciu a slovník `d`, v ktorom sa ako kľúče vyskytujú niektoré premenné, ktoré sú aj v `f` – nie nutne všetky, a ako hodnoty v `d` sú funkcie v prefixovej reprezentácii. Ako návratovú hodnotu vráti `compose` zloženú funkciu v prefixovej reprezentácii.

2.1 Vzor

```
>>> compose(1,{})
1
>>> compose(1,{"x":["+","y","z"]})
1
>>> compose("x",{"x":["+","y","z"]})
['+', 'y', 'z']
>>> compose("x",{})
'x'
>>> compose("x",{"y":100})
'x'
>>> compose("x",{"x":100})
100
>>> compose("x",{"x":"y"})
'y'
>>> compose(["+","x",100],{"x":"y"})
['+', 'y', 100]
>>> compose(["+","x",100],{"x":100})
['+', 100, 100]
>>> compose(["+","x","x"],{"x":100})
['+', 100, 100]
>>> compose(["+","x","x"],{"x":"y"})
['+', 'y', 'y']
>>> compose(["+","x","x"],{"x":["+","y",1]})
['+', ['+', 'y', 1], ['+', 'y', 1]]
>>> compose(["+","x","z"],{"x":["+","y",1]})
['+', ['+', 'y', 1], 'z']
>>> compose(["+","x","z"],{"x":["+","y",1],"z":["*","x",17.5]})
['+', ['+', 'y', 1], ['*', 'x', 17.5]]
>>> compose(["+","x","x"],{"x":["+","y",1],"z":["*","x",17.5]})
['+', ['+', 'y', 1], ['+', 'y', 1]]
>>> compose(["+",["*","z","x"],"x"],{"x":["+","y",1],"z":["*","x",17.5]})
['+', ['*', ['*', 'x', 17.5], ['+', 'y', 1]], ['+', 'y', 1]]
```

3 Prirovnajte (9 bodov)

Toto zadanie vyžaduje zamyslenie sa, ale nie je to nič strašné.

Napíšte funkciu `mymatch(f,pat)` kde `f`, `pat` sú funkcie v prefixovej reprezentácii. Môžete predpokladať, že v `pat` sa premenné neopakujú To znamená, že

```
["*","x",["/",10,["+","y","x"]]]
```

je neprípustná hodnota `pat`, lebo `'x'` je tam dva razy.

Funkcia má vrátiť slovník `d` taký, že `compose(pat,d) == f`. Ak taký slovník neexistuje, má vrátiť hodnotu `None`.

3.1 Vzor

```
>>> print mymatch(1,1)
{}
>>> print mymatch(1,2)
None
>>> print mymatch("x",2)
None
>>> print mymatch(2,"x")
{'x': 2}
>>> print mymatch("y","x")
{'x': 'y'}
>>> print mymatch(["+","z","w"],"x")
{'x': ['+', 'z', 'w']}
>>> print mymatch(["+","z",10.5],["+","x","y"])
{'y': 10.5, 'x': 'z'}
>>> print mymatch(["*","z",10.5],["+","x","y"])
None
 >>> print mymatch(["+","z",10.5],["+",["*","a","b"],"y"])
None
>>> print mymatch(["+",["*",1,"x"],10.5],["+",["*","a","b"],"y"])
{'a': 1, 'y': 10.5, 'b': 'x'}
>>> print mymatch(["+",["*",1,"x"],10.5],["+",["-","a","b"],"y"])
None
>>> print mymatch(["+",["*",1,"x"],10.5],["+","z","y"])
{'y': 10.5, 'z': ['*', 1, 'x']}
```

4 Užitočné fragmenty kódu

4.1 Zlučovanie slovníkov cez `dict.update`

```
>>> d={"x":1,"y":["+","z",2]}
>>> d1={"v":7,"w":["-","9","z"]}
>>> d.update(d1)
>>> print d
{'y': ['+', 'z', 2], 'x': 1, 'w': ['-', '9', 'z'], 'v': 7}
>>>
```

4.2 Súbežná iterácia pomocou `zip`

```
>>> l1=['z',2]
>>> l1=["z",2]
>>> l2=["w","x"]
>>> zip(l1,l2)
[('z', 'w'), (2, 'x')]
>>> for e1,e2 in zip(l1,l2):
...  print "e1=",e1,"e2=",e2
...
e1= z e2= w
e1= 2 e2= x
>>>
```