Skúšobné zadanie na druhé cvičenie

15. mája 2015

1 Príprava

Vaše riešenie bude mať súbor s menom `vase_priezvisko.py`. Bude obsahovať iba definície funkcií. Nechajte v ňom aj funkciu `myeval` z domácej prípravy (ak ju máte).

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 Derivujte (8 bodov)

Napíšte funkciu, `myderive(f,var)`, ktorá vráti prefixovú reprezentáciu parciálnej derivácie funkcie `f` danej prefixovou reprezentáciou.

```
>>> myderive(1,"x")
0
>>> myderive("y","x")
0
>>> myderive("x","x")
1
>>> myderive("y","x")
0
>>> myderive(["-",2,"x"],"x")
['-', 0, 1]
>>> myderive(["*",2,"x"],"x")
['+', ['*', 0, 'x'], ['*', 2, 1]]
>>> myderive(["*","x","x"],"x")
['+', ['*', 1, 'x'], ['*', 'x', 1]]
>>> myderive(["*","x","x"],"y")
['+', ['*', 0, 'x'], ['*', 'x', 0]]
>>> myderive(["*",["-","x",1],"x"],"x")
['+', ['*', ['-', 1, 0], 'x'], ['*', ['-', 'x', 1], 1]]
>>> myderive(["+","x","x"],"x")
['+', 1, 1]
>>> myderive(["+","y","x"],"x")
['+', 0, 1]
>>> myderive(["/","x","y"],"y")
['/', ['-', ['*', 0, 'y'], ['*', 'x', 1]], ['*', 'y', 'y']]
```

Samozrejme, implementácia bude opäť rekurzívna. Uvedomte si, že stačí implementovať známe pravidlá o derivácii konštanty, premennej a aritmetických operácií nad funkciami. Netrápte sa nad tým, že vám vznikajú komplikované funkcie. Tým sa zaoberá ďalšia časť zadania.

Ďalej je dobré si uvedomiť, že napríklad pravidlo o násobení konštantou nemusíte implementovať, pretože vyplýva z pravidla o derivácii konštanty a pravidla o derivácii súčinu funkcií.

Skôr ako začnete komplikovane pomocou pomocných premenných konštruovať zoznamy cez `extend` a `append`, si uvedomte, že v Pythone môžete bez problémov používať výrazy ako `['*',`*`niečo`*`,`*`niečo_iné`*`]` priamo v kóde.

3 Zjednodušte (7 bodov)

Keďže myderive vracia dosť zbytočne zložité funkcie, pre rozumnú použiteľnosť treba výsledok zjednodušiť. Napíšte funkciu `mysimplify(f)`, ktorá dostane funkciu v prefixovej reprezentácii a vráti tiež funkciu v prefixovej reprezentácii, ale

jednoduchšiu.

Implementujte pritom tieto transformácie.

- $x + 0, 0 + x \mapsto x$
- $x.1, 1.x \mapsto x$
- $x.0, 0.x \mapsto 0$
- $x/1 \mapsto x$
- výraz bez premenných (ergo konštanta) $\mapsto$ jeho hodnota.

Posledný bod v našich podmienkach sa najlepšie vyrieši cez volanie `myeval(f,{})`.

Uvedomte si, že funkcia bude opäť rekurzívna; najprv sa zjednodušia podvýrazy a až na zjednodušené podvýrazy sa aplikujú pravidlá o „lokálnom zjednodušovaní".

```
>>> mysimplify(["*",1,1])
1
>>> mysimplify(["*",1,"x"])
'x'
>>> mysimplify(["*",["*",1,1],"x"])
'x'
>>> mysimplify(["*",["*",0,1],"x"])
0
>>> myderive(["*",["+",1,"x"],"x"],"x")
['+', ['*', ['+', 0, 1], 'x'], ['*', ['+', 1, 'x'], 1]]
>>> mysimplify(myderive(["*",["+",1,"x"],"x"],"x"))
['+', 'x', ['+', 1, 'x']]
>>> myderive(["*",["+",1,"x"],"x"],"y")
['+', ['*', ['+', 0, 0], 'x'], ['*', ['+', 1, 'x'], 0]]
>>> mysimplify(myderive(["*",["+",1,"x"],"x"],"y"))
0
>>> myderive(["/",["+",1,"x"],"x"],"x")
['/', ['-', ['*', ['+', 0, 1], 'x'], ['*', ['+', 1, 'x'], 1]], ['*', 'x', 'x']]
>>> mysimplify(myderive(["/",["+",1,"x"],"x"],"x"))
['/', ['-', 'x', ['+', 1, 'x']], ['*', 'x', 'x']]
```

4 Odovzdávanie

Cez bitbucket.