

Diff:

Differences between given skeleton and solution

In order to make the sample solution easier to understand, the differences between it and the given skeleton source code were highlighted with the help of the program `diff`.

Legend:

- Gray: unchanged text (only excerpts).
- Green: new lines
- Yellow: changed lines
- Red: deleted lines

Note: Files not listed have not been changed.

This document was created with the help of [diff2html](#) erstellt.

```
diff -u ../course09-symbolic-computation/exercise/code/lagrange.py ../course09-symbolic-computation/exercise/solution/lagrange.py
```

```
../course09-symbolic-computation/exercise/code/lagrange.py
```

```
../course09-symbolic-computation/exercise/solution/lagrange.py
```

```
3 import sys
4
5
6 ## this file contains a skeleton and has to be amended
7 # the variable XXX is only a placeholder and has to be replaced appropriately in each case
8
9 # task 1
10 params = sp.symbols("m1, m2") # incomplete (parameters are missing)
11 m1, m2 = params
12
13 sys.exit() # move this line further down as you proceed with the exercise
14
15
16
17 # task 2
18 t = sp.Symbol("XXX") # create the symbol for the time
19 xt = Function("x")(t) # x(t)
20 phit = Function(XXX)(XXX) # phi(t)
21
22 # task 3 (construct four time derivatives)
23 xdt = xt.diff(t)
24 ## ...
25 xddt = xt.diff(t, 2)
26 ## ...
27
28
29 # task 4
30
31 # auxiliary quantities
32 x2t = XXX
33 y2t = XXX
34
35 x2dt = XXX.diff(t)
36 ## ...
37
38 # task 5
39 # kinetic energy
40 T = (m1*xdt**2 + XXX)/2
41
42 # potential energy
43 U = XXX
44
45 L = T - U # Lagrange funktion
46
47
48
49
50
51
52
53
54
55
56 # auxiliary expressions:
57 L_d_x = L.diff(xt)
58 L_d_phi = XXX
59
60 L_d_xd = XXX
61 L_d_phid = XXX
62
63 # task 7
64 DL_d_xd = XXX.diff(t)
```

```
3 import sys
4
5
6
7 # task 1
8 params = sp.symbols("m1, m2, l, g")
9 m1, m2, l, g = params
10
11
12
13 # task 2
14 t = sp.Symbol("t") # create the symbol for the time
15 xt = Function("x")(t) # x(t)
16 phit = Function("phi")(t) # phi(t)
17
18 # task 3 (construct four time derivatives)
19 xdt = xt.diff(t)
20 phidt = phit.diff(t)
21 xddt = xt.diff(t, 2)
22 phiddt = phit.diff(t, 2)
23
24
25 # task 4
26
27 # auxiliary quantities
28 x2t = xt + l*sin(phit)
29 y2t = -l*cos(phit)
30
31 x2dt =x2t.diff(t)
32 y2dt =y2t.diff(t)
33
34 # task 5
35 # kinetic energy
36 T = (m1*xdt**2 + m2*(x2dt**2 + y2dt**2))/2
37
38 # potential energy
39 U = y2t*g*m2
40
41 L = T - U # Lagrange funktion
42
43
44
45
46
47
48
49
50
51
52 # auxiliary expressions:
53 L_d_x = L.diff(xt)
54 L_d_phi = L.diff(phit)
55
56 L_d_xd = L.diff(xdt)
57 L_d_phid = L.diff(phidt)
58
59 # task 7
60 DL_d_xd = L_d_xd.diff(t)
```

```

65 DL_d_phid = XXX
66
67
68 # task 8
69 F = sp.Symbol("F") # external force (translatoric)
70
71 # right hand side of the equations of motion (left hand side is zero)
72 Eq1 = XXX - XXX - F
73 Eq2 = XXX
74
75 # useful for debugging: pretty printing
76 # sp.pprint(Eq1)
77 # sp.pprint(Eq2)
78
79
80 # task 9
81 # list of accelerations
82 acc = [xddt, XXX]
83
84 # solve equations for acceleration symbols
85 res = sp.solve([XXX, XXX], acc)
86
87 # task 10
88
89 msg = f"\nThe variable `res` is of type: {XXX} and has the following value:\n"
90 print(msg)
91 sp.pprint(res)
92
93 xdd_expr = res[xddt]
94 phidd_expr = XXX
95
96
97 # task 11
98
99
100
101
102
103
104
105
106
107
108
109
110
111 (xt, x), (phit, phi)]
112
113 # step 2:
114 params_values = [(m1, 0.8), XXX]
115
116
117 # perform substitution and save result in variables
118 xdd_expr_num = xdd_expr.subs(rplmts+params_values)
119 phidd_expr_num = XXX
120
121
122 # preparation done; now we can create the python functions
123
124 # generation of the Python functions using sp.lambdify
125 xdd_fnc = sp.lambdify([x, phi, xd, phid, F], xdd_expr_num, modules="numpy")
126 phidd_fnc = sp.lambdify(XXX)
127
128
129

```

```

61 DL_d_phid = L_d_phid.diff(t)
62
63
64 # task 8
65 F = sp.Symbol("F") # external force (translatoric)
66
67
68
69 # right hand side of the equations of motion (left hand side is zero)
70 Eq1 = DL_d_xd - L_d_x - F
71 Eq2 = DL_d_phid - L_d_phi
72
73
74 # useful for debugging: pretty printing
75 sp.pprint(Eq1)
76 sp.pprint(Eq2)
77
78
79 # task 9
80 # list of accelerations
81 acc = [xddt, phiddt]
82
83 # solve equations for acceleration symbols
84 res = sp.solve([Eq1, Eq2], acc)
85
86 # task 10
87
88 msg = f"\nThe variable `res` is of type: {type(res)} and has the following value:\n"
89 print(msg)
90 sp.pprint(res)
91
92 xdd_expr = res[xddt]
93 phidd_expr = res[phiddt]
94
95
96 # task 11
97
98
99
100
101
102
103
104
105
106
107
108
109
110 (xt, x), (phit, phi)]
111
112 # step 2:
113 params_values = [(m1, 0.8), (m2, 0.3), (l, 0.5), (g, 9.81)]
114
115
116 # perform substitution and save result in variables
117 # note: a sum of lists is a new list containing all elements
118 xdd_expr_num = xdd_expr.subs(rplmts+params_values)
119 phidd_expr_num = phidd_expr.subs(rplmts+params_values)
120
121
122 # preparation done; now we can create the python functions
123
124 # generation of the Python functions using sp.lambdify
125 xdd_fnc = sp.lambdify([x, phi, xd, phid, F], xdd_expr_num, modules="numpy")
126 phidd_fnc = sp.lambdify([x, phi, xd, phid, F], phidd_expr_num, modules="numpy")
127
128
129

```