Mutable defaults

September 17, 2024

1 Mutable arguments and defaults

In general, it is bad form to mutate arguments passed into a function (we want our functions to be pure), but sometimes it is necessary because we want to avoid copying lots of data, or because we are integrating with legacy code.

Default values for parameters get created at the same time as the function object. In fact, they are stored in the function object. This has surprising effects for Python beginners, when they set a mutable object as a default value.

2 Never use mutable defaults

[1]:
```
def myfun(mylist=[]):
    mylist.append(1)
    print(mylist)

myfun()
myfun()
```

```
[1]
[1, 1]
```

2.1 Do this instead

[2]:
```
def myfun2(mylist=None):
    if not mylist:
        mylist = [1]
    print(mylist)

myfun2()
myfun2()
```

```
[1]
[1]
```

2.2 Function as argument with lambda

[33]:
```
costs = [('b', '11'), ('a', '10.3'), ('x', '-2')]
sorted(costs, key=lambda x: float(x[1]))
```

[33]:
```
[('x', '-2'), ('a', '10.3'), ('b', '11')]
```

2.3 Function as argument without lambda

[34]:
```
def mysort(x):
    return (float(x[1]))

sorted(costs, key=mysort)
```

[34]:
```
[('x', '-2'), ('a', '10.3'), ('b', '11')]
```