Udacity: Machine Learning Engineer

Contents

1 Software Engineering Fundamentals

1.1 Software Engineering Practices

Modular Code: putting functions into separate files to be imported into workspace.

Refactoring: restructuring your code to improve its internal structure, without changing its external functionality. This means cleaning and modularizing your program after it is working.

Optimization: we want to write efficient code, so this can be either fast execution or taking up less space in memory. We also want to *vectorize* our code for speed and amount of coding used.

```
for book in recent_books:
  if book in coding_books:
    recent_coding_books.append(book) # 16.63 sec

recent_coding_books = np.intersect1d(recent_books, coding_books) # 0.035 sec
recent_coding_books = set(recent_books).intersection(coding_books) # 0.0097 sec

for cost in gift_costs:
  if cost < 25:
    total_price += cost * 1.08  # 5.55 sec

total_price = np.sum(gift_costs[gift_costs < 25] * 1.08) # 0.084 sec

```

Git Branches: to switch to a branch in a repository you use *git checkout (branchname)*.
To create and switch to a new branch you use *git checkout -b (newbranch)*.
When in main branch, merge another branch by using *git merge -no-ff (branchname)*.

Previous Code: to see previous commits use *git log*.
Using the commit message number, open the code using a new branch *git checkout (commit#)*.

Unit Testing: pytest is a tool we can use to make sure our function is outputting correctly. We can create a test file starting with *test_* and we get a . if we pass and an F if we fail.

Test Driven Deployment: writing tests before you write the code that's being tested. Your test would fail at first, and you'll know you've finished implementing a task when this test passes.

1.2 Object-Oriented Programming