

Clean coding and computational reproducibility

Russ Poldrack
Stanford University

To err is human

Professional software developers make 1 - 50
errors per 1000 lines of code

How many errors do we make as amateur
developers?

How likely are they to impact our scientific
conclusions?

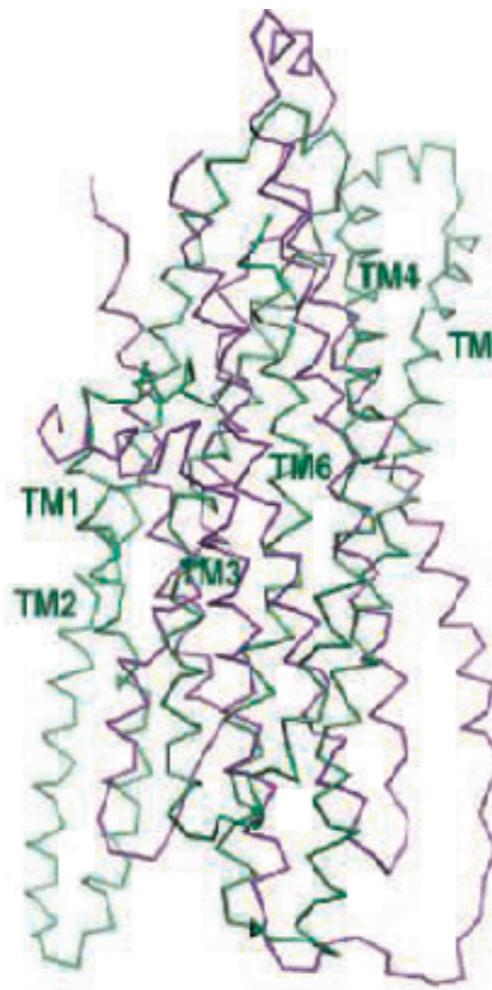
The impact of software errors

**Structure of MsbA from *E. coli*:
A Homolog of the Multidrug
Resistance ATP Binding Cassette
(ABC) Transporters**

Geoffrey Chang* and Christopher B. Roth

Multidrug resistance (MDR) is a serious medical problem and presents a major challenge to the treatment of disease and the development of novel therapeutics. ABC transporters that are associated with multidrug resistance (MDR-ABC transporters) translocate hydrophobic drugs and lipids from the inner to the outer leaflet of the cell membrane. To better elucidate the structural basis for the "flip-flop" mechanism of substrate movement across the lipid bilayer, we have determined the structure of the lipid flippase MsbA from *Escherichia coli* by x-ray crystallography to a resolution of 4.5 angstroms. MsbA is organized as a homodimer with each subunit containing six transmembrane α -helices and a nucleotide-binding domain. The asymmetric distribution of charged residues lining a central chamber suggests a general mechanism for the translocation of substrate by MsbA and other MDR-ABC transporters. The structure of MsbA can serve as a model for the MDR-ABC transporters that confer multidrug resistance to cancer cells and infectious microorganisms.

www.sciencemag.org SCIENCE VOL 293 7 SEPTEMBER 2001

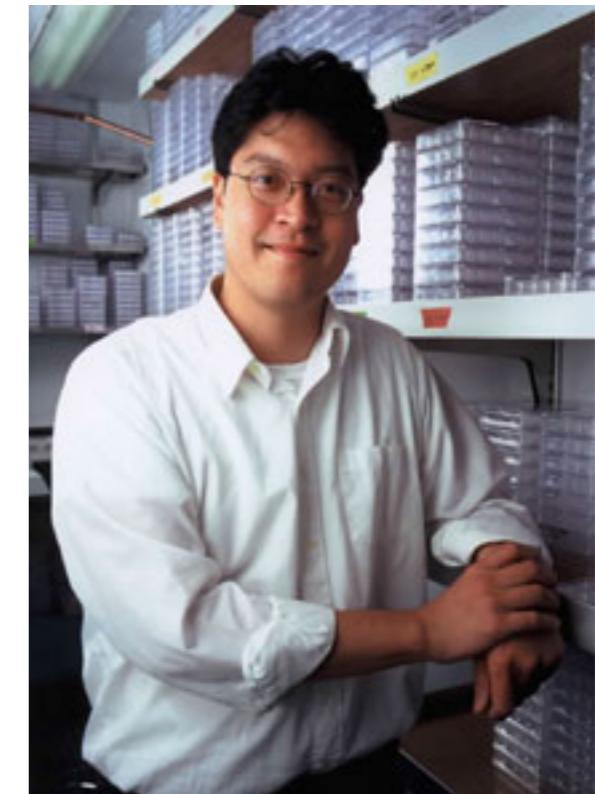


**Structure of the ABC Transporter
MsbA in Complex with ADP·Vanadate
and Lipopolysaccharide**

Christopher L. Reyes and Geoffrey Chang*

Select members of the adenosine triphosphate (ATP)-binding cassette (ABC) transporter family couple ATP binding and hydrolysis to substrate efflux and confer multidrug resistance. We have determined the x-ray structure of MsbA in complex with magnesium, adenosine diphosphate, and inorganic vanadate ($MgADP\cdot V_i$) and the rough-chemotype lipopolysaccharide, Ra LPS. The structure supports a model involving a rigid-body torque of the two transmembrane domains during ATP hydrolysis and suggests a mechanism by which the nucleotide-binding domain communicates with the transmembrane domain. We propose a lipid "flip-flop" mechanism in which the sugar groups are sequestered in the chamber while the hydrophobic tails are dragged through the lipid bilayer.

13 MAY 2005 VOL 308 SCIENCE www.sciencemag.org



Geoffrey Chang

**X-ray Structure of the EmrE
Multidrug Transporter in Complex
with a Substrate**

Owen Pornillos, Yen-Ju Chen, Andy P. Chen, Geoffrey Chang*

EmrE is a prototype of the Small Multidrug Resistance family of efflux transporters and actively expels positively charged hydrophobic drugs across the inner membrane of *Escherichia coli*. Here, we report the x-ray crystal structure, at 3.7 angstrom resolution, of one conformational state of the EmrE transporter in complex with a translocation substrate, tetraphenylphosphonium. Two EmrE polypeptides form a homodimeric transporter that binds substrate at the dimerization interface. The two subunits have opposite orientations in the membrane and adopt slightly different folds, forming an asymmetric antiparallel dimer. This unusual architecture likely confers unidirectionality to transport by creating an asymmetric substrate translocation pathway. On the basis of available structural data, we propose a model for the proton-dependent drug efflux mechanism of EmrE.

23 DECEMBER 2005 VOL 310 SCIENCE www.sciencemag.org

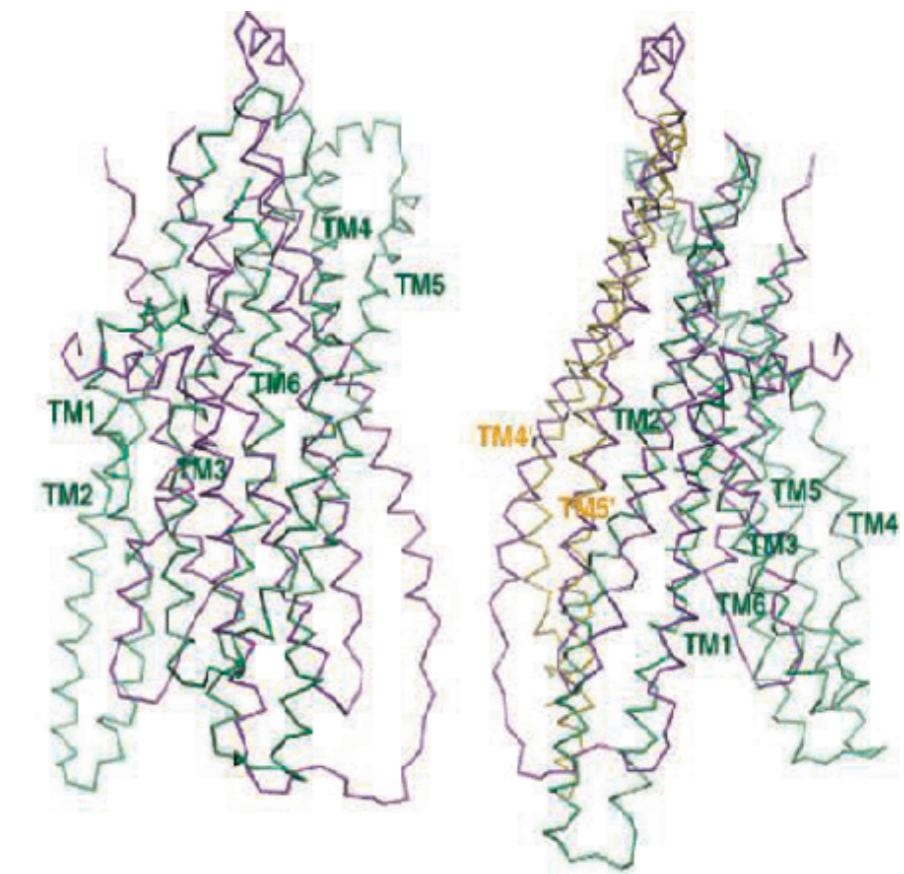
The impact of software errors

Retraction

WE WISH TO RETRACT OUR RESEARCH ARTICLE “STRUCTURE OF MsbA from *E. coli*: A homolog of the multidrug resistance ATP binding cassette (ABC) transporters” and both of our Reports “Structure of the ABC transporter MsbA in complex with ADP•vanadate and lipopolysaccharide” and “X-ray structure of the EmrE multidrug transporter in complex with a substrate” (1–3).

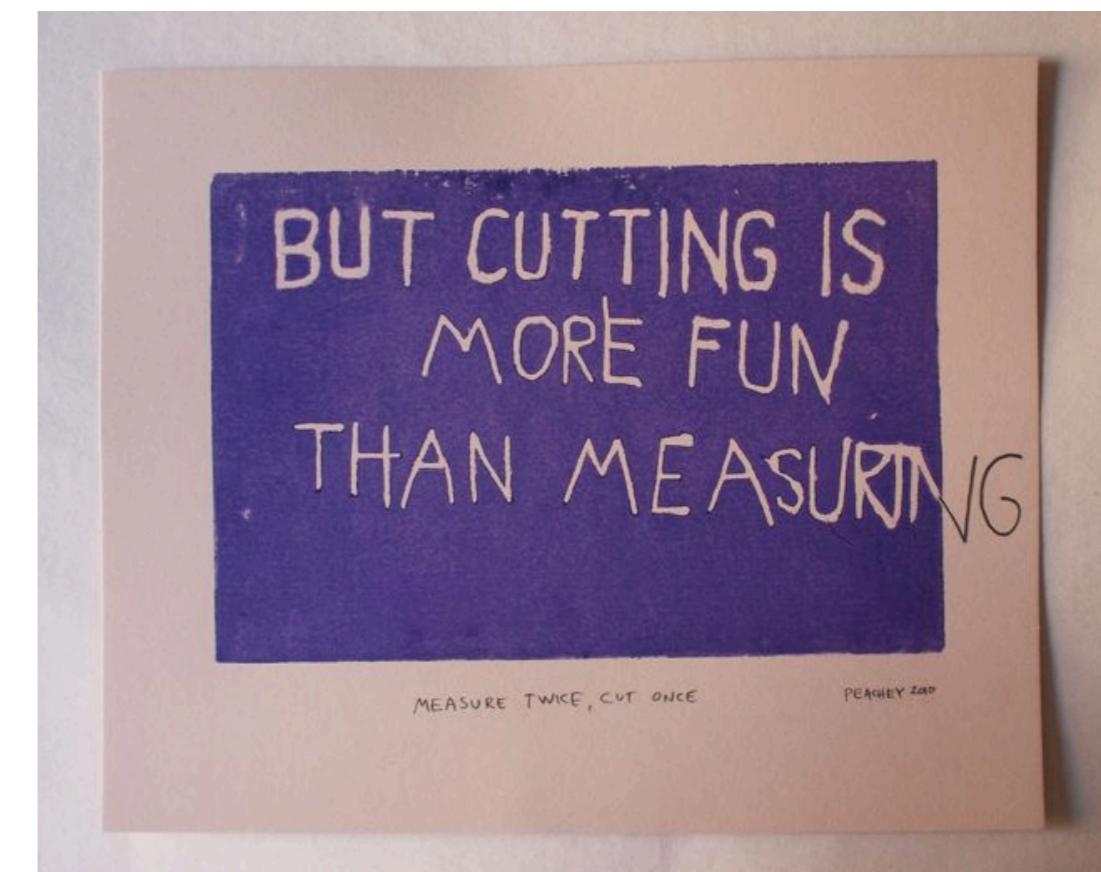
The recently reported structure of Sav1866 (4) indicated that our MsbA structures (1, 2, 5) were incorrect in both the hand of the structure and the topology. Thus, our biological interpretations based on these inverted models for MsbA are invalid.

An in-house data reduction program introduced a change in sign for anomalous differences. This program, which was not part of a conventional data processing package, converted the anomalous pairs (I^+ and I^-) to (F^- and F^+), thereby introducing a sign change. As the diffraction data collected for each set of MsbA crystals and for the EmrE crystals were processed with the same program, the structures reported in (1–3, 5, 6) had the wrong hand.



Coding for reproducibility

- Learning clean and reproducible coding practices increases efficiency in the long run
 - “As in real carpentry — the kind done with lumber — the time saved by measuring carefully before cutting a piece of wood is much greater than the time that measuring takes.” - <https://swcarpentry.github.io/python-novice-inflammation/10-defensive/index.html>

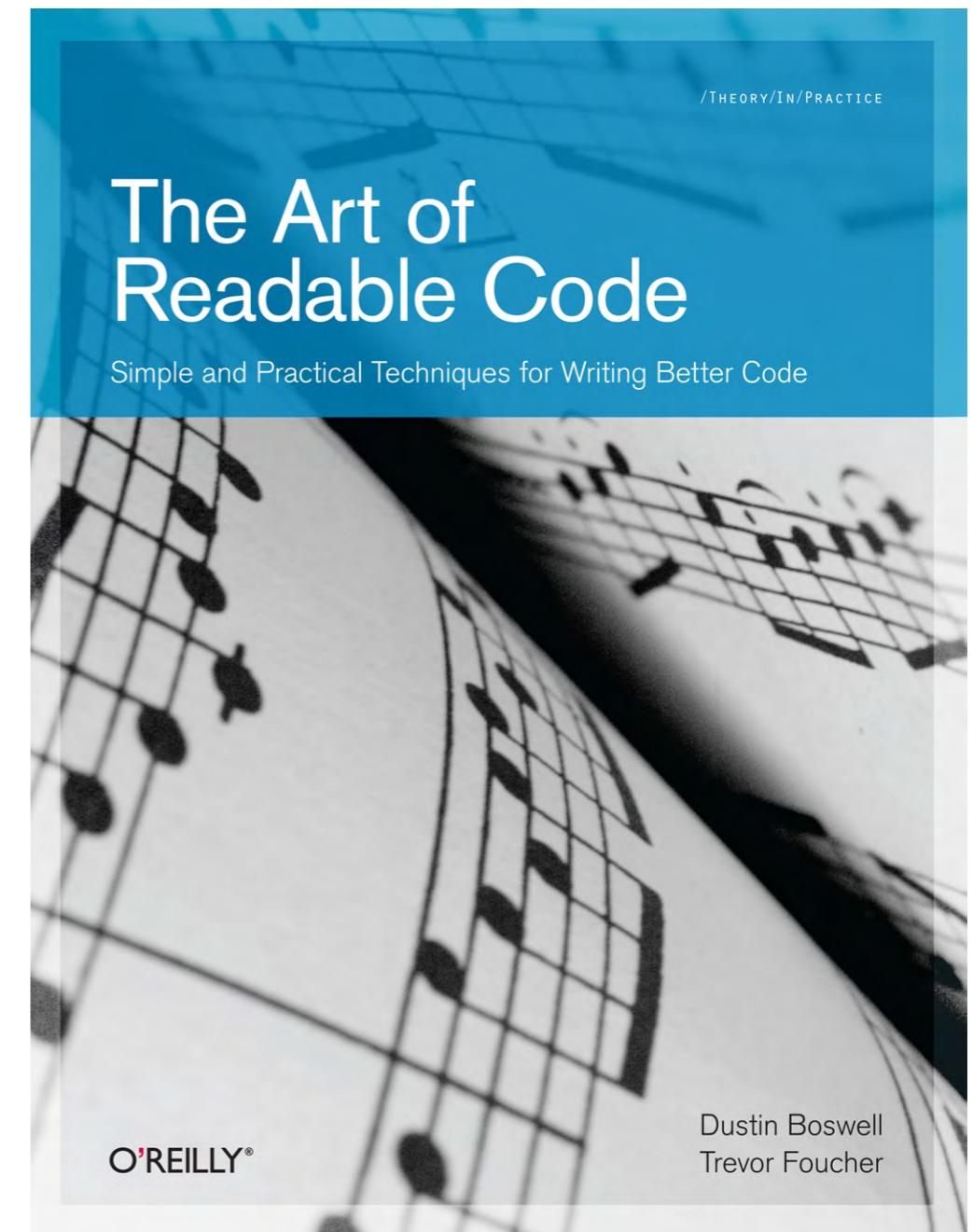


Good Coding Practices

- Code understandably
- Code defensively
- Code portably
- Automate as much as possible

Coding understandably

- You should try to minimize the time it would take for someone else to understand your code
 - Where “someone else” also includes your future self!



Naming

- Names should be as informative as possible

```
m = {'Jets': ['Riff', 'Tony', 'Diesel'], 'Sharks': ['Bernardo', 'Chino', 'Pepe']}
c = ['Doc', 'Shrank', 'Krupke', 'Riff', 'Tony', 'Diesel', 'Bernardo', 'Chino', 'Pepe']
```

```
for k in range(len(c)):
    for i in m:
        for j in range(len(m[i])):
            if m[i] == c[j]:
                print('%s is in %s' % (m[i], c[j]))
```

incorrect index

```
club_members = {
    'Jets': ['Riff', 'Tony', 'Diesel'],
    'Sharks': ['Bernardo', 'Chino', 'Pepe']
}
cast_members = ['Doc', 'Shrank', 'Krupke', 'Riff', 'Tony', 'Diesel', 'Bernardo', 'Chino', 'Pepe']

for cast_member in cast_members:
    for club in club_members:
        if cast_member in club_members[club]:
            print(f'{cast_member} is in {club}')
```

Follow standard style guidelines

- PEP8 naming conventions
 - Never use the characters 'l' (lowercase letter el), 'O' (uppercase letter oh), or 'I' (uppercase letter eye) as single character variable names.
 - In some fonts, these characters are indistinguishable from the numerals one and zero. When tempted to use 'l', use 'L' instead.
 - Variable and function names should be lowercase, with words separated by underscores as necessary to improve readability (“snake case”)

Avoid “magic numbers”

- Any numeric values should be assigned to a clearly named variable

```
c = 0
for i in range(1000):
    if my_number_generator() > 1.96:
        c += 1
print(c/1000)
```

```
exceedence_counter = 0
num_simulation_runs = 1000
threshold = 1.96

for run in range(num_simulation_runs):
    if my_number_generator() > threshold:
        exceedence_counter += 1
print(exceedence_counter/num_simulation_runs)
```

what if this line
occurred much
later in the code?
would you remember
to change it?



Using a style checker (flake8 plugin for VSCode)

```
● 3  from numpy import *
4
5  ✓ def my_data_generator(datasize,mean=None,sd=0):
6      if mean == None: mean = 0
7      data=random.randn( 100 )*sd + mean
8      l=mean(data)
9      return(l)
10
11
12  print(my_data_generator(8,0,1))
```

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL Filter. E.g.: text, **/*.ts, !**/node_modules/**

- example_for_flake8.py ~/Dropbox/code/clean_coding 17
 - ✗ 'from numpy import *' used; unable to detect undefined names flake8(F403) [3, 1]
 - ✗ expected 2 blank lines, found 1 flake8(E302) [5, 1]
 - ✗ missing whitespace after ';' flake8(E231) [5, 31]
 - ✗ missing whitespace after ';' flake8(E231) [5, 41]
 - ✗ comparison to None should be 'if cond is None:' flake8(E711) [6, 13]
 - ✗ multiple statements on one line (colon) flake8(E701) [6, 20]
 - ✗ missing whitespace around operator flake8(E225) [7, 9]
 - ✗ 'random' may be undefined, or defined from star imports: numpy flake8(F405) [7, 10]
 - ✗ whitespace after '(' flake8(E201) [7, 23]
 - ✗ whitespace before ')' flake8(E202) [7, 27]
 - ✗ missing whitespace around arithmetic operator flake8(E226) [7, 29]
 - ✗ ambiguous variable name 'l' flake8(E741) [8, 5]
 - ✗ missing whitespace around operator flake8(E225) [8, 6]
 - ✗ missing whitespace after ';' flake8(E231) [12, 26]
 - ✗ missing whitespace after ';' flake8(E231) [12, 28]
 - ⚠ no newline at end of file flake8(W292) [12, 32]
 - ⓘ Inline variable that is immediately returned, Use x is None rather than x == None sourcery(refactor) [6, 1]

Commenting is not a magic balm for bad code

- “Don’t comment bad code — rewrite it” (Kernighan & Plaugher)
- Comments should help the reader to know as much as the writer knows about the code
- “Clear and expressive code with few comments is far superior to cluttered and complex code with lots of comments” (Robert Martin, *Clean Code*)

What *not* to comment

- Ugly or indecipherable code
 - Refactor so that the code is understandable

```
# function to create a random normal variate
# takes mean and SD as arguments
def crv(a, b):
    ...
```

- instead, use understandable names:

```
def create_random_normal_variate(mean, sd):
    ...
```

What *not* to comment

- The obvious

```
# looping over states
for state_index, state in enumerate(states):
    ...
```

```
# function to create a random normal variate
def create_random_normal_variate(mean, sd):
    ...
```

What *not* to comment

- Historical information

```
# RP changed on 8/12/2020 to include sd argument
def create_random_normal_variate(mean, sd):
    ...
```

- Should instead be tracked in version control system

```
git commit -m"added sd argument to create_random_normal_variate"
```

What to comment

- Intention (aka “Director Commentary”)

```
# using a numpy array here rather than  
# a pandas DataFrame for performance reasons
```

- Known flaws or TODO items

```
# TODO: Currently this will fail if x == pi  
# - should include code to address that edge case
```

- Comments on your constant values

```
# Using 500 bootstrap replicates based on  
# Efron & Tibshirani 1993
```

Coding defensively

- Assume that errors will occur, and create code that is either:
 - Robust to errors
 - Detects and announces errors loudly

Assertions

- Assert that some statement must be true
 - Otherwise raise an Exception

```
In [5]: spike_count = -5
```

```
In [6]: assert spike_count >= 0
```

```
AssertionError                                     Traceback (most recent call last)
<ipython-input-6-3e7acfef5ed4> in <module>
----> 1 assert spike_count >= 0
```

```
AssertionError:
```

CORRECTION

Correction: The Role of Conspiracist Ideation and Worldviews in Predicting Rejection of Science

Stephan Lewandowsky, Gilles E. Gignac, Klaus Oberauer

The dataset included two notable age outliers (reported ages 5 and 32757).

Specifically, the statement on page 9 “age turned out not to correlate with any of the indicator variables” is incorrect. It should read instead “age correlated significantly with 3 latent indicator variables (Vaccinations: .219, $p < .0001$; Conservatism: .169, $p < .001$; Conspiracist ideation: -.140, maximum likelihood $p < .0001$, bootstrapped $p = .004$), and straddled significance for a fourth (Free Market: .08, $p = .05$).”

```
In [1]: age=32757
```

```
In [2]: assert age>12 and age<120
```

```
AssertionError                                     Traceback (most recent call last)
<ipython-input-2-37de876b5fda> in <module>()
----> 1 assert age>12 and age<120
```

```
AssertionError:
```

Software testing

- Important functions should be tested
 - Determine what your code should do, and write a test to make sure that it does that thing properly
 - Determine the cases in which your code should raise exceptions, and make sure it properly raises them
- Test-driven development:
 - Add a test
 - Run the test and see if it fails
 - Add code needed to pass the test
 - Repeat until all needed functionality is present

Testing using pytest

```
from rtanalysis.rtanalysis import RTAnalysis

def test_rtanalysis_smoke():
    rta = RTAnalysis()
    assert rta is not None
```

```
pytest_tutorial % python -m pytest tests/test_1_smoketest.py
===== test session starts =====
platform darwin -- Python 3.8.3, pytest-5.4.1, py-1.8.1, pluggy-0.13.1
rootdir: /Users/poldrack/Dropbox/code/pytest_tutorial
plugins: cov-2.10.0
collected 1 item
```

```
rtanalysis/test_1_smoketest.py . [100%]
```

```
===== 1 passed in 0.25s =====
```

Code portably

- Don't hard-code machine-dependent details into code
 - File paths, system names, credentials
 - Instead, either use environment variables or configuration files
 - Be sure not to check configuration files into github!
 - use the `.gitignore` file
 - JSON files are very convenient
 - easily save and retrieve Python dicts

Automating your workflow

- Goal: be able to re-run the entire workflow on a new machine with a single command
- Simple tool: UNIX make

Makefile:

```
all: setup analysis

setup:
    python do_setup.py

analysis:
    python run_analysis_step1.py
    python run_analysis_step2.py
```

```
(py38) % make setup
python do_setup.py
doing setup
```

```
(py38) % make analysis
python run_analysis_step1.py
doing analysis step 1
python run_analysis_step2.py
doing analysis step 2
```

```
(py38) % make all
python do_setup.py
doing setup
python run_analysis_step1.py
doing analysis step 1
python run_analysis_step2.py
doing analysis step 2
```

Learning what not to do

Python Anti-Patterns

Search docs

- ✓ Correctness
- ✗ Maintainability
- ✖ Readability
- ✖ Security
- ✖ Performance
- ✖ Django

[Documentation](#) » The Little Book of Python Anti-Patterns

[View page source](#)

The Little Book of Python Anti-Patterns

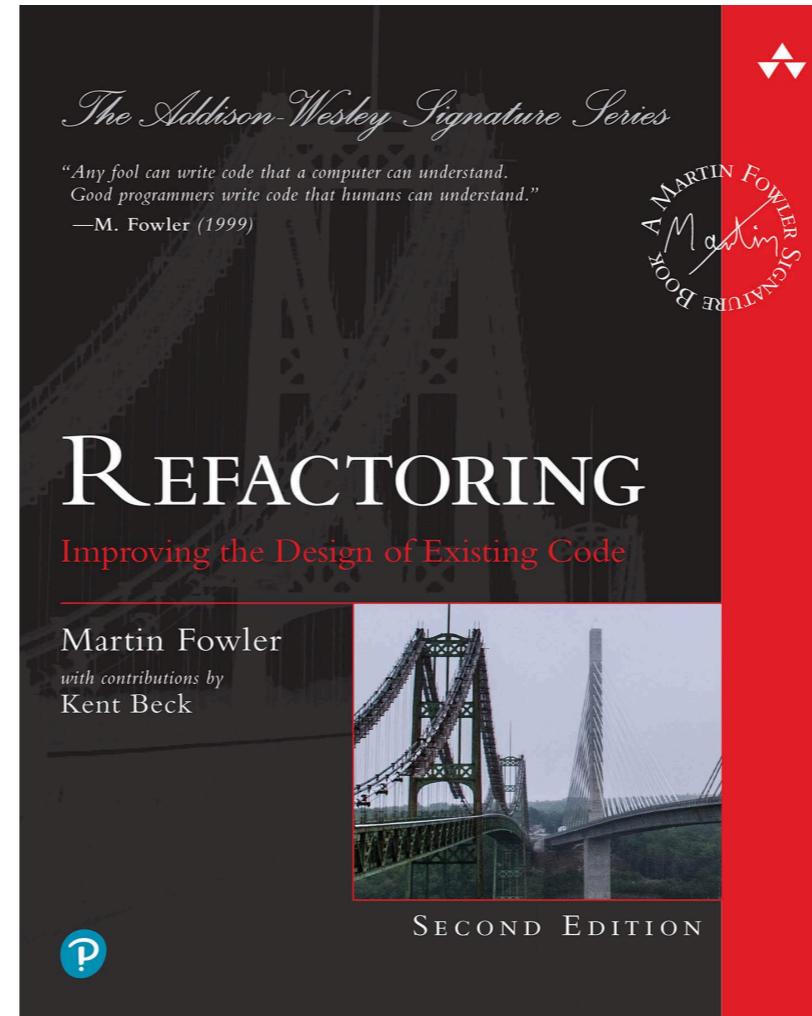


Welcome, fellow Pythoneer! This is a small book of Python **anti-patterns** and **worst practices**.

Learning about these anti-patterns will help you to avoid them in your own code and make you a better programmer (hopefully). Each pattern comes with a small description, examples and possible solutions. You can check many of them for free against your project at [QuantifiedCode](#).

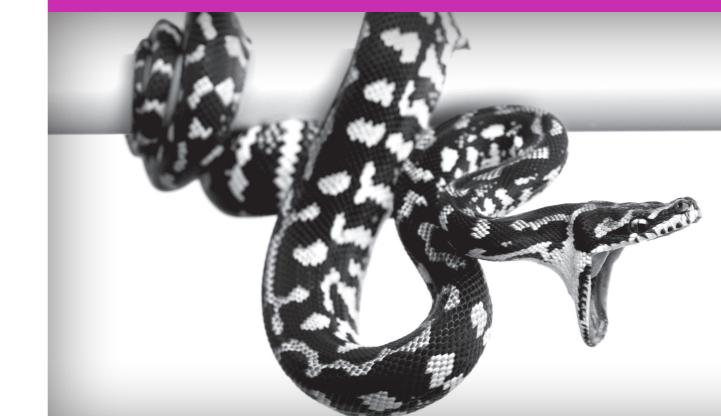
<https://docs.quantifiedcode.com/python-anti-patterns/>

- “anti-patterns”
 - commonly used bad solutions to common problems
- “code smells”
 - Patterns in software that signal potential problems



O'REILLY®

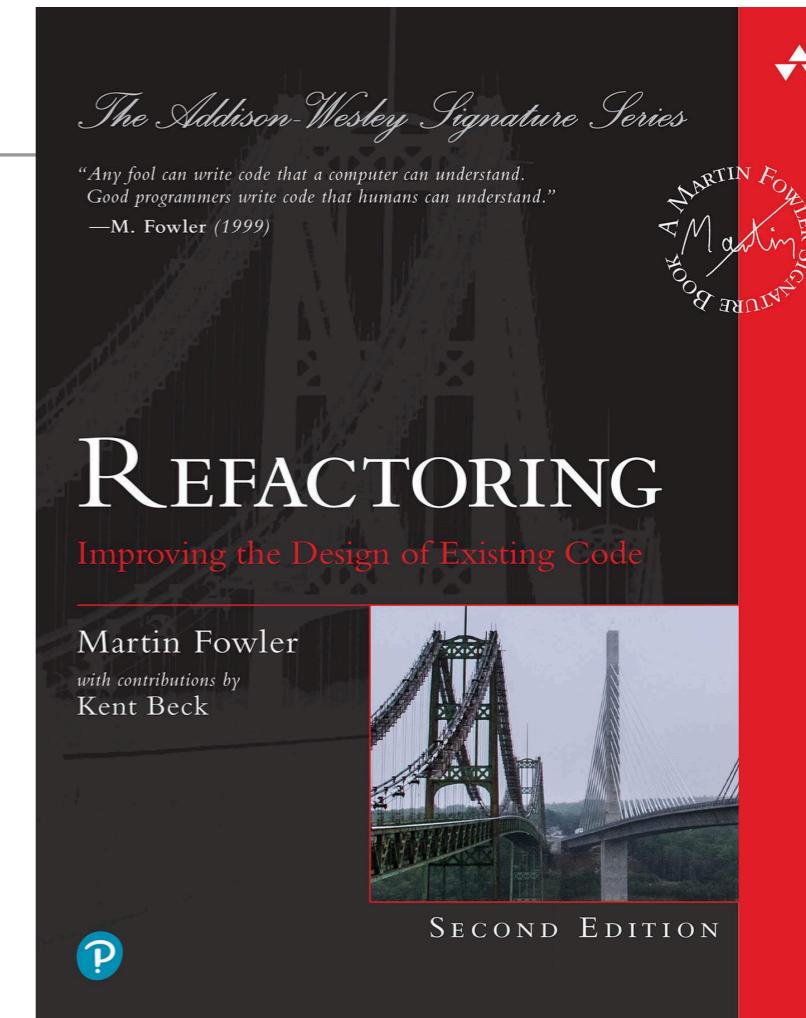
How to Make Mistakes in Python



Mike Pirnat

Common “bad smells” in code

- Mysterious name
- Duplicated code
- Long function
- Long parameter list
- Global data
- Mutable data
- Requirement for changes in multiple places
- Repeated switches
- Comments
 - “comments aren’t a bad smell; indeed they are a sweet smell. The reason we mention comments here is that comments are often used as a deodorant. It’s surprising how often you look at thickly commented code and notice that the comments are there because the code is bad.”



The DRY (Don't Repeat Yourself) principle

- Creating multiple variables with numbers or names in them is a bad practice ('anti-pattern')

```
tes
test1 = 'stroop'
test2 = 'flanker'
test3 = 'nback'
```

```
res
result1 = analyze(test1)
result2 = analyze(test2)
result3 = analyze(test3)
```



“code smell”

YAGNI: You Ain't Gonna Need It

- You should focus on solving today's needs
 - Predictions about future needs are often wrong
 - Adding unnecessary complexity makes code harder to maintain
- You should focus on building code that is easy to extend in the future
 - e.g. use lookup tables (dictionaries) rather than hardcoded values

SRP: single responsibility principle

- A function or class should do only one thing
 - Makes it much easier to understand
 - Also easier to modify without side effects
- When using SRP, most functions should be very short

The opposite of SRP: The “God Object”

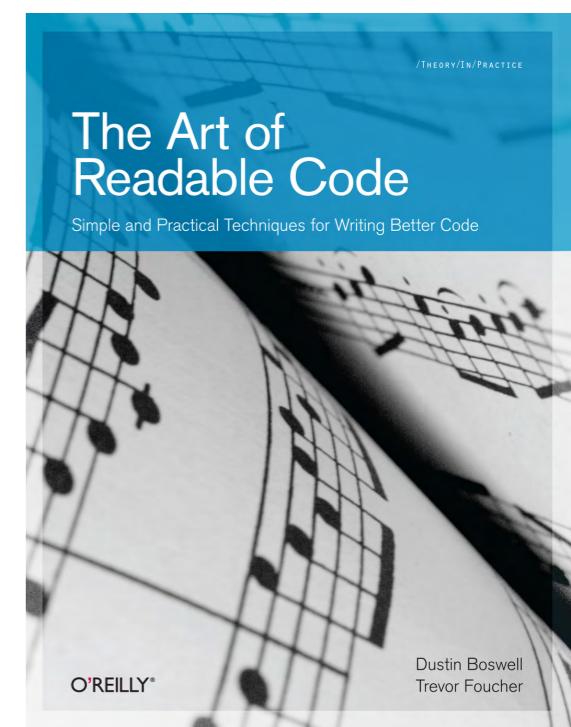
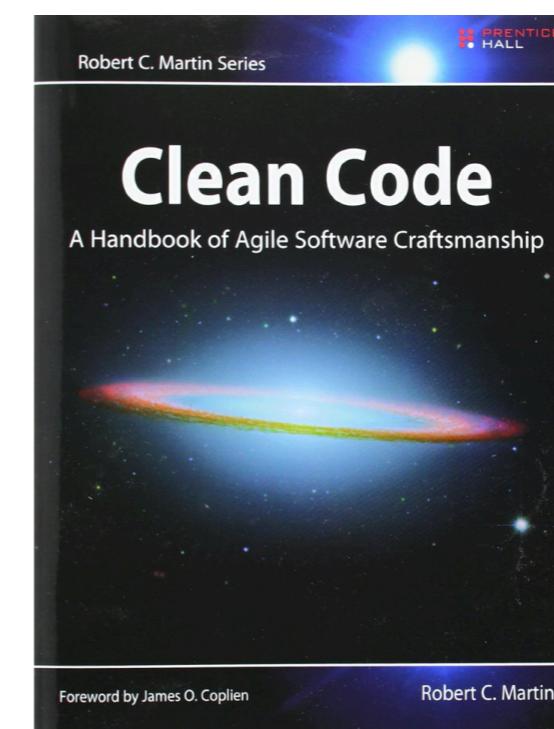
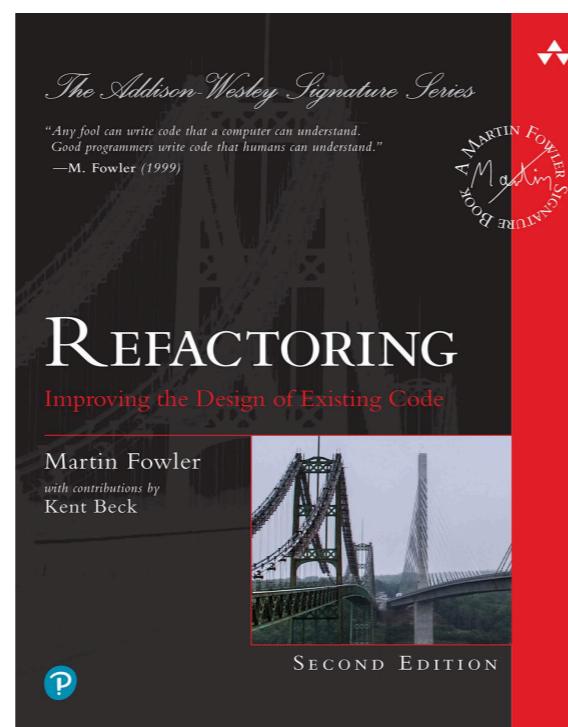
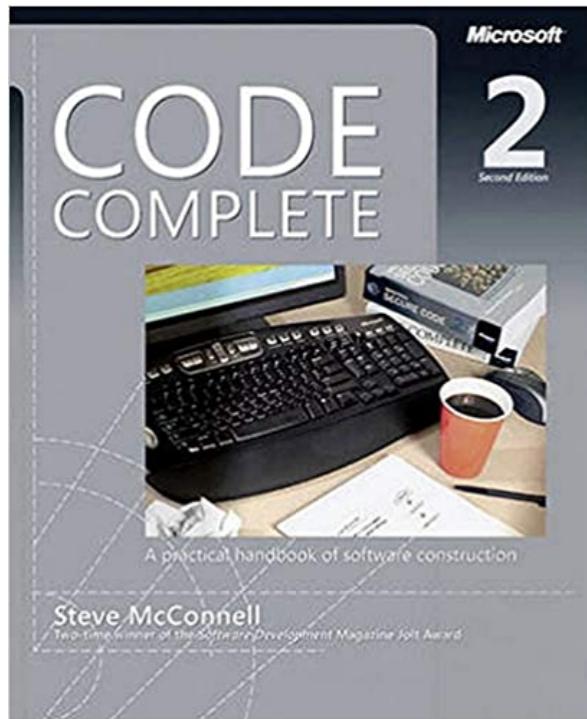
```

404  class Narps(object):
405      """
406      main class for NARPS analysis
407      """
408
409      def __init__(self, basedir, metadata_file=None,
410                  verbose=False, overwrite=False,
411                  dataurl=None, testing=False):
412          self.basedir = basedir
413          self.dirs = NarpsDirs(basedir, dataurl=dataurl,
414                                testing=testing)
415          self.verbose = verbose
416          self.teams = {}
417          self.overwrite = overwrite
418          self.started_at = datetime.datetime.now()
419          self.testing = testing
420
421
422      for teamID in self.complete_image_sets['thresh']:
423          self.teams[teamID] = NarpsTeam(
424              teamID,
425              info['teams'][teamID]['NV_collection_id'],
426              info['dirs'],
427              verbose=self.verbose)
428          self.teams[teamID].jsonfile = info[
429              'teams'][teamID]['jsonfile']
430          self.teams[teamID].images = info[
431              'teams'][teamID]['images']
432          self.teams[teamID].image_json = info[
433              'teams'][teamID]['image_json']
434          self.teams[teamID].input_dir = info[
435              'teams'][teamID]['input_dir']
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043

```

Implementing good coding practices

- Code review
 - You write code, someone else reviews it
 - Good for highlighting readability problems
- Pair programming
 - work together with someone else synchronously
 - VSCode has a Live Share mode (ala Google Docs)
- Learn about practical software engineering practices



Breakout group exercise

- Look at this code:
 - https://github.com/poldrack/clean_coding/blob/master/example1.py
 - (don't peek at the other versions of this code in the repo!)
- With your group, spend 10 minutes going over the code, with the following questions in mind:
 - What are the different parts of the code doing?
 - What code smells do you sense?
 - How might you refactor it to address these?
- Appoint a group spokesperson who can report back if called upon