CAMBRIDGE
Cambridge University Press
978-1-108-78987-5 — How to Write Good Programs
Perdita Stevens
Index
More Information

Index

A **bold** page number indicates where a term is defined.

abstract syntax tree, 105
abstraction, 29, 141
  see also model
agile, 64, 159, 198
algorithm, 29, 147, 148, 196
Alice, 44
arguments, **25**, 28
  functions as, 45
  order of, 108
  type of, 41, 108
assert, 71
assignment, 131
  vs. comparison, 126
Atom, 18
autocompletion, **52**, 90
autosave, 57

backups, 65
bar, *see* metasyntactic variable
BASIC, 44, 125
baz, *see* metasyntactic variable
BlueJ, 44, 54
breakpoint, 111
bug, 32, **101**, 190, 191
  after removing, 124
  avoiding, 138
  avoiding recurrence of, 77
  in compiler, 109
  removing, 122
  the Lauren bug, 78
  see also debugging
build, 51, 53

C, 35, 45, 189
C$^\sharp$, 35
C++, 35, 190
camel case, 89
change, 141, 144, 197
checklist, 127
cloud, 66
code
  commented-out, 62
  completion, **52**, 90
  dead, **64**
  line length, 99
  reputable body of, 46, 63, 94, 99
  spaghetti, **98**, 122
  unreachable, **64**
code sense, 3, 133
coding, **4**
coding dojo, 153
coding interview, 148
command line, 15, 49
comment, 27, 70, 85–88
commenting-out, **62**
comparison
  of booleans, 126
  of objects, 130
  vs. assignment, 126
compiler, 13, 35
  bug, 109
  incremental, 51
computational complexity, 148
content assist, **52**, 90
contract, **88**

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-1-108-78987-5 — How to Write Good Programs
Perdita Stevens
Index
More Information

crash, 118
currying, 23

data science, 192
database, 189
debugger, 111
debugging, 101–133
  by rewriting, 112
  cardboard, 120
  crash, 118
  interactive, 111
  non-termination, 118
  rubber duck, 120
  using print statements, 110
defensive programming, **132**
demonstration, 175
dependency, 37, 42, 51, 138, 163, 188
design, 134, 141, 144, 183, 197
  notes, 29, 88, 152
design pattern, 197
development
  iterative, 30
  test-driven, **79**, 151
Dijkstra, Edsger, 82
directory, 17
disk, 56
domain concept, 144
domain-specific language, 198
duplication
  eliminating, 136

Eclipse, 51, 53, 54, 90, 92
editor, 16, 49, 55, 90
  dumb, 49, 53
  mode, 49, 95
  vs. word processor, 16
effect, 25
efficiency, 134, 145
  maintainer, 146
  Pareto, 146
  programmer, 146
  space, 146
  time, 146
  user, 146
Emacs, 18, 49, 58, 90, 95
embedded systems, 190
encapsulation, 45, 114
enterprise systems, 190
error, **104**
  compared with warning, 104
  compile-time, 103
  non-termination, 118
  null pointer exception, 119, 130
  off-by-one, 130
  run-time, 118
  type, 107
  see also bug
error indicator, 51
error message, 103
  recurrent, 160
ethics, 10
exam, 179
exception, 118
exercises, 157

failure, *see* bug
fault, *see* bug
Fizz Buzz, 19, 20, 25, 28, 138, 153
folder, 17
foo, 32
for loops, 127
Fortran, 190
framework, **76**, 190, 192
  logging, 111
  testing, 75
froboz, 32
function, 42
  pure, 25

Git, 60
GitHub, 53, 61, 66
graphical user interface, 188

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-1-108-78987-5 — How to Write Good Programs
Perdita Stevens
Index
More Information

habit, 55, 65, 66, 122, 127, 129, 131, 132, 153
hackathon, 194
Hamilton, Margaret, 78, 196
Haskell, 45, 190
Haskell example, 15, 18, 22, 26, 40, 80, 107, 108, 113, 129, 139
Hello World, 38
Hoare, Tony, 183
hobbits, 155
Hollywood Principle, 76
Hopper, Grace, 101
hubris, 155

IDE, *see* integrated development environment
idiom, 39, 46, **97**, 148
impatience, 155
integrated development environment, **14**, 18, 50–55, 90, 95
IntelliJ IDEA, 53
interactive prompt, 15, 111
interpreter, 13, 34
invariant, 88

Java, 45, 190
Java annotations, 130
Java example, 6, 22, 27, 39, 72, 73, 84, 85, 87, 91, 93, 94, 106, 114, 115, 117, 125, 130, 131, 137, 138, 140
JavaScript, 34, 190
JUnit, 75
Jupyter notebook, 16

kata, **153**
keyboard shortcut, 51
Knuth, Don, 82

layout, 52, 92
laziness, 155
lexing, 105
library, **43**, 51, 162, 190, 192
  standard, **43**, 47, 162
linking, 36
Lisp, 192
logging, 111
Logo, 44

machine learning, 192
magic constant, **139**
maintainability, 134
maintenance, 134
MATLAB, 191
memory, 56
metasyntactic variable, 32
method, **25**, 42, 72
  main, 72
Milner, Robin, 35
minimal non-working example, **164**
ML, 35
MNWE, **164**
model, **197**
model-driven engineering, 198
Moore's Law, **145**
Moore, Gordon, 145
multiple choice, 185
mung, 32
mutable state, 45

name, 89–92, 142, 144
  for constant, 140
NetBeans, 53
non-termination, 118, 128
Notepad, 17
Notepad++, 18
NPE, **119**, 130
null pointer exception, **119**, 130

Objective-C, 35
off-by-one error, **130**
open-source, 43, 61, 191, 192, 194
operating system, 15, 49

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-1-108-78987-5 — How to Write Good Programs
Perdita Stevens
Index
More Information

pair programming, **159**
paradigms, 45
parsing, **105**
Perl, 34, 74, 191
PHP, 34, 191
pointer, 189
polymorphism, 41
positive bias, 77
post-condition, 88
pre-condition, 88
print statements, 110
program, **12**
  good enough, 152
  Hello World, 38
programming, **4**
  defensive, 132
  extreme, 198
  functional, 45
  imperative, 45
  logic, 45
  object-oriented, 45, 143
  pair, 159
programming language theory, 198
project, 50, 173
Prolog, 45, 191
Python, 191
Python example, 5, 15, 26, 28, 38, 68, 72, 97, 123, 128, 129, 142, 143, 151

quick fixes, 52

R, 192
Racket, 192
ratchet technique, 58
RCS, 60
recursion, 129
refactoring, 92, **121**, 134–152
reflective writing, 177
regression, **77**
  see also testing
repetitive strain injury, 51
REPL, 15
requirements, 196

Scheme, 192
scope, **42**
Scratch, 14, 44
scripting language, 44, 74, 191
segmentation fault, 118
semantics, **21**, 199
sexism, 11
skeleton, 28
snake case, 89
software engineering, **4**, 134, 196
software process, 198
space character, 95
spaghetti code, 122, 163
specification, 28, **68**, 69, 80
stack overflow, 118
stack trace, **119**
standard out, 68, 110
static type-checking, 189
stdout, 68, 110
structure, 41, 99, 139, 141, 144, 152
Sublime, 18
Subversion, 60
SVN, 60
syntax, **21**
syntax error, 21
syntax highlighting, 49

tab character, 95
TDD, **79**, 85, 151
team working, 174, 198
template file, 18
test, **69**
  system, **74**
  unit, **73**
test-driven development, **79**, 85, 151
testing, 68–82, 196
  property-based, 79
  quotations about, 81
  regression, 77, 102, 109

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-1-108-78987-5 — How to Write Good Programs
Perdita Stevens
Index
More Information

testing main, 72
TextEdit, 18
Turing complete, 186, 187
Turing machine, **187**
Turing, Alan, 187
Turing-computable functions, 187
type-checking, **37**
  static, 189
types, 37–41, 199
TypeScript, 190

UML, 197
undo, 55

version control, 53, 59–61, 66
vi, 18
Visual Basic, 44
Visual Studio, 54

Wall, Larry, 154
warning, **104**
web applications, 190
while loops, 127
whitespace, 92, 95
workflow, 50

Y2K, 188

© in this web service Cambridge University Press www.cambridge.org