Writing Beautiful Code

Anand Chitipothu

Who is speaking?

Anand Chitipothu

@anandology

Advanced programing courses @pipalacademy

Building data science platform @rorodata

quality without a name

Programs must be written for people to read, and only incidentally for machines to execute.

- Structure and Interpretation of Computer Programs (The Wizard Book)

Choose Meaningful Names

Two hard things in computer science are cache invalidation and naming things.

- Phil Karlton

Avoid Generic Names

```
tmp
tmp2
manager
data
```

Avoid Abbreviations

```
ucf = UpperCaseFormatter()
ba = BankAccount()

formatter = UpperCaseFormatter()
account = BankAccount()
```

Avoid using datatype as name

```
sum(list)
count_words(string)


sum(numbers)
count_words(sentence)
```

Nouns & Verbs

Use nouns for variables and classes.

```
size, price, Task, Scheduler, Bank Account
```

Use verbs for functions.

```
get_file_size, make_account, deposit
```

Use plural for a list

largest_line(**lines**)

files = os.listdir(directory)

file = os.listdir(directory)

for lines in open(filename).readlines():

    sum += int(lines)

Reserve Loop Indexes

Use i, j only as loop indexes.

```
for i in range(10): print i
```

```
for i in numbers: result += i
for n in numbers: result += n
```

Example 1

Can you improve this?

```
def get_data(x, y):
    z = []
    for i in x:
        z.append(i[y])
    return z
```

Example 1

```
def get_column(dataset, col_index):
    column = []
    for row in dataset:
        column.append(row[col_index])
    return column
```

Similar names

Never use similar names for completely different datatypes.

```
a1 = [1, 2, 3]
a2 = len(x)


values = [1, 2, 3]
n = len(x)
```

Comments

Don’t say the obvious

```
# increments x by 2
x = x + 2

# compensate for border on both the sides
x = x + 2
```

Explain why you made that choice

```
# The following is an optimization to saves
# lot of memcache calls. Handle with care!
...
```

Document special cases

```
# -- XXX -- Anand - Sep 2015 --
# UTF-conversion was failing for a chinese
# user for reasons I couldn't understand.
# Added "ignore" as second argument to handle
# that temporarily.
name = name.encode("utf-8", "ignore")
```

Make Comments Redundant

```
# find length of the longest line
n = max([len(line) for line in lines])
```

```
n = len(longest(lines))
```

Make Comments Redundant

```
# process documents
...
# upload them to search engine
...

docs = process_documents(...)
search_engine_submit(docs)
```

Program Organization

Divide & Conquer

Split the program into small independent modules and functions.

The 7 ± 2 Rule

The number of objects an average human can hold in working memory is 7 ± 2.

- Miller's Law

Avoid Duplication

```
def add(input_data):
    try:
      x = int(input_data['x'])
    except ValueError:
      raise Exception("Invalid int value for x")

    try:
      y = int(input_data['x'])
    except ValueError:
      raise Exception("Invalid int value for y")

    return x+y
```

Avoid Duplication - generalize instead

```
def get_int(dictionary, key):
    try:
     return int(dictionary[key])
    except ValueError:
     raise Exception("Invalid int value for {}".format(key))

def add(input_data):
    x = get_int(input_data, "x")
    y = get_int(input_data, "y")
    return x+y
```

Avoid too many nested levels

```
def update_post(...):
    post = get_post(..)
    if action == 'update-title':
        if title == '':
            ...
        else:
            ...
    elif action == "add-tag":
            ...
```

Avoid too many nested levels

```
def update_post(...):
    post = get_post(..)
    if action == "update-title":
        update_post_title(...)
    elif action == "add-tag":
        update_post_add_tag(...)
```

Handle errors separately

```
def get_user(email):
    if valid_user(email):
        if is_user_blocked(email):
            return Exception("Account is blocked")
        else:
            query = "...."
            row = db.select(query).first()
            return User(row)
    else:
        raise Exception("Invalid email")
```

Handle errors separately

```
def get_user(email):
    if not valid_user(email):
        raise ValueError("Invalid email")
    if is_email_blocked(email):
        raise Exception("Account blocked")

    query = "...."
    row = db.select(query).first()
    return User(row)
```

Suppress the implementation details

```
def main():
    filename = sys.argv[1]
    words = read_words(filename)
    freq = wordfreq(words)
    print_freq(freq)
```

Summary

- Choose meaningful variable names
- Use comments when required
- Split the program into small independent modules & functions
- Avoid duplication
- Suppress implementation details
- Always optimize for readability

“A program should be light and agile, its subroutines connected like a string of pearls. The spirit and intent of the program should be retained throughout. There should be neither too little nor too much, neither needless loops nor useless variables, neither lack of structure nor overwhelming rigidity.”

- The Tao of Programming

“A program should be light and agile, its subroutines connected like a string of pearls. **The spirit and intent of the program should be retained throughout.** There should be neither too little nor too much, neither needless loops nor useless variables, neither lack of structure nor overwhelming rigidity.”

- The Tao of Programming

“A program should be light and agile, its subroutines connected like a string of pearls. The spirit and intent of the program should be retained throughout. **There should be neither too little nor too much, neither needless loops nor useless variables, neither lack of structure nor overwhelming rigidity.”**

- The Tao of Programming

Happy Coding!

@anandology