THE BEGINNER BOOKS OF...

NAMING THINGS

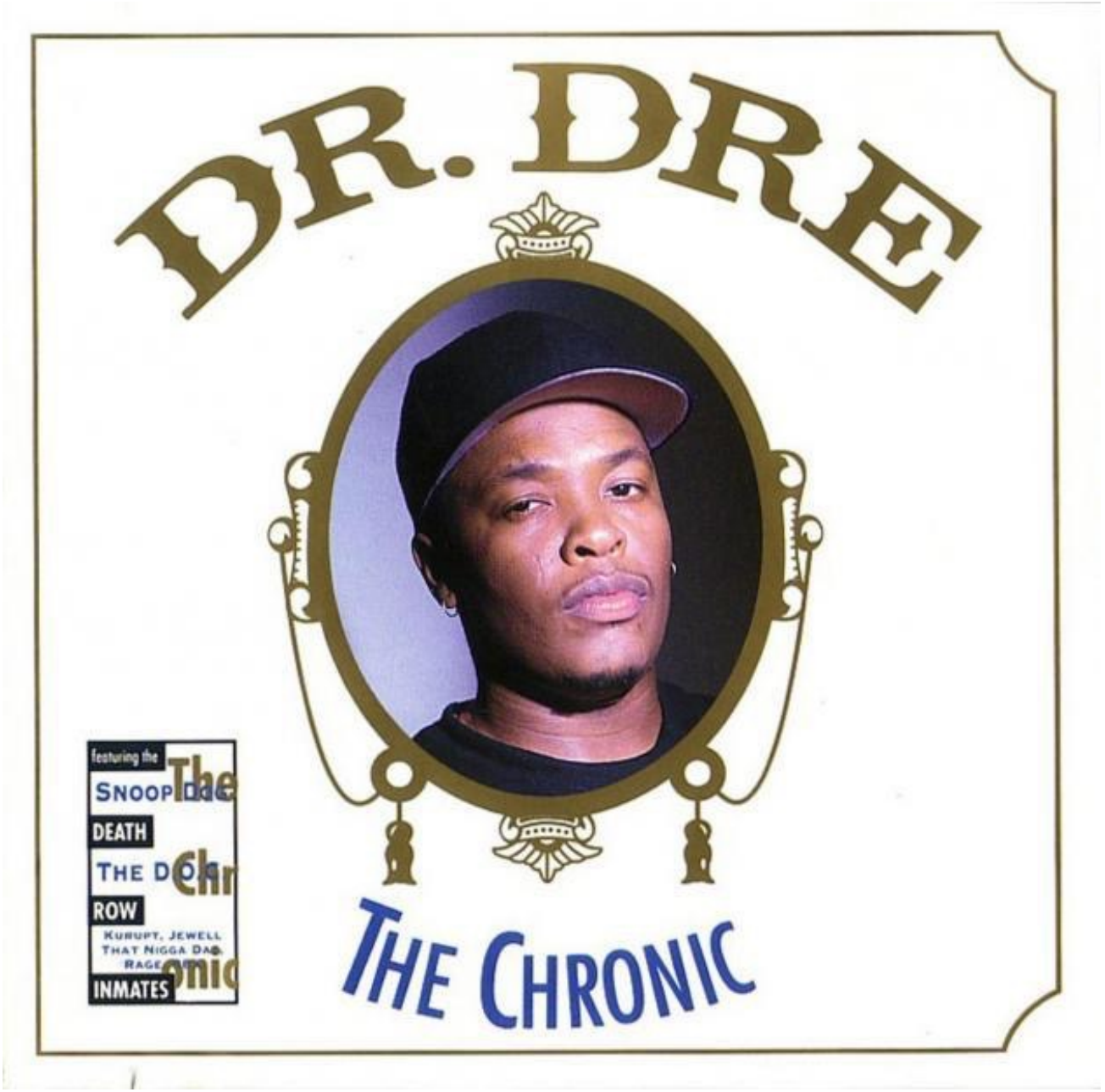
GITHUB: HTTPS://GITHUB.COM/JEREMYTEDWARDS/BEGINNER-BOOKS

HTTPS://BLOGS.MSDN.MICROSOFT.COM/ERICLIPPERT/2004/06/14/READING-CODE-IS-HARD/

I DON'T THINK IT'S TOO MUCH OF A STRETCH TO SAY THAT THE HARDEST PART OF CODING IS NOT WRITING CODE, BUT READING IT... READING CODE IS HARD.

Eric Lippert

DR. OF NAMING THINGS…NOT THIS GUY.



DR. OF NAMING THINGS.... THIS GUY.

HTTP://HAACKED.COM/ARCHIVE/2007/04/19/WRITE-READABLE-CODE-BY-MAKING-ITS-INTENTIONS-CLEAR.ASPX/

"STRIVE TO BE THE DR. SEUSS OF WRITING CODE."

Phil Haack

(works at github, has been to burning man, and lives in Bellevue, WA)

SAMPLE CODE

- file: **CIH.py**

```
class Box(object):
    hook = True

    def FunInABox(self, Thing1, Thing2):
        return Thing1, Thing2

    @staticmethod
    def shakeHands(self):
        Sally = True
        Me = True
        Fish = "Put them out!"
        return Sally, Me, Fish

    @staticmethod
    def PutAway(self, thing1, thing2):
        fish = ""
        if Mother:
            fish = "Oh Dear"
            catchWithNet(Me)
        return thing1, thing2, fish
```

HTTPS://WWW.PYTHON.ORG/DEV/PEPS/PEP-0008/#NAMING-CONVENTIONS

PEP-0008

- Modules (your files) should have short, all-lowercase names. It is usually preferable to stick to 1 word names.
- Class names should normally use the CamelCase convention.
- Use the function naming rules: lowercase with words separated by underscores as necessary to improve readability.

PEP-0008

- Constants are usually defined on a module level and written in all capital letters with underscores separating words. Examples include MAX_OVERFLOW and TOTAL .

HTTPS://WWW.PYTHON.ORG/DEV/PEPS/PEP-0008/#NAMING-CONVENTIONS

PEP-0008

- Never use 'l' (lowercase letter el), 'O' (uppercase letter oh), or 'I' (uppercase letter eye) as single character variable names. When tempted to use 'l', use 'L' instead.
- _single_leading_underscore : weak "internal use" indicator. Don’t use this outside of this class.
- single_trailing_underscore_ : used by convention to avoid conflicts with Python keyword, e.g.

```
Tinker.Toplevel(master, class_='ClassName')
```

SAMPLE CODE

- file: **CIH.py**

```
class Box(object):
    hook = True

    def FunInABox(self, Thing1, Thing2):
        return Thing1, Thing2

    @staticmethod
    def shakeHands(self):
        Sally = True
        Me = True
        Fish = "Put them out!"
        return Sally, Me, Fish

    @staticmethod
    def PutAway(self, thing1, thing2):
        fish = ""
        if Mother:
            fish = "Oh Dear"
            catchWithNet(Me)
        return thing1, thing2, fish
```

SAMPLE CODE

- file: **cat_in_hat.py**

```
class BigRedBox(object):
    _HAS_A_HOOK = True

    def fun_in_a_box(self, thing_one, thing_two):
        return thing_one, thing_two

    @staticmethod
    def things_shake_hands(self, thing_one, thing_two):
        shook_with_sally = True
        shook_with_me = True
        fish_says = "Put them out!"
        return shook_with_sally, shook_with_me, fish_says

    @staticmethod
    def things_put_away(self, thing_one, thing_two):
        fish_says = ""
        if is_mother_coming:
            fish_says = "Oh dear!"
            catch_with_net(me)
        return thing_one, thing_two, fish_says
```

SAMPLE CODE

file: **green_egg_ham_raw.py**

```
times_asked = 0

def ask_if_you_like_green_eggs_and_ham(where="green eggs and ham?"):
    my_question = "do you like " + where
    response = input(my_question)
    return response

def main():
    eggs_are_liked = ask_if_you_like_green_eggs_and_ham()
    times_asked = 1
    questions = {
        "q1": "them here or there?",
        "q2": "them in a house with a mouse?",
        "q3": "them in a box with a fox?",
        "q4": "them on a train?",
        "q5": "them in the dark?",
        "q6": "them on a boat?",
    }
    for i in questions:
        if eggs_are_liked is not True:
            times_asked += 1
            my_question_is = questions[i]
            eggs_are_liked = ask_if_you_like_green_eggs_and_ham(my_question_is)
        else:
            break
    print("Times asked: ", times_asked)
```

AVOIDING THE ANNOYING

- Don’t use a single letter for iterables, it’s makes it hard to search code without getting false positives.
- Avoid making names that are prefixes of other names like my_question and then later use my_question_is.
- Shadowing is evil. Please don’t do it.
  (where inner variable shadows an outer local variable e.g. count)

HTTP://VISUALGIT.READTHEDOCS.ORG/EN/LATEST/PAGES/NAMING_CONVENTION.HTML

NAMING

- Avoid using names that are too general or too wordy. Strike a good balance between the two.

  Bad: my_menu, my_dictionary, ask_if_you_like_green_eggs_and_ham()

  Good: menu_options, word_definitions, ask_do_you_like_where()

- When using CamelCase names, capitalize all letters of an abbreviation (e.g. HTTPServer)

HTTPS://GITHUB.COM/LIONHEART/STYLE-GUIDE/BLOB/MASTER/GUIDES/PYTHON/NAMING%20THINGS.MD

FILTERING

- If there is any sort of filtering on or action applied to a boxed type, you generally want to prepend the name with an adjective.

```
sold_out_products = getProducts.filter()

booked_hotels = getHotels.filter()
```

ORDINALS AND NUMBERS

- If naming a value representing a count of objects, preface the variable with num.

```
num_times_asked = get_all_timesAsked()
```

HTTPS://GITHUB.COM/LIONHEART/STYLE-GUIDE/BLOB/MASTER/GUIDES/PYTHON/NAMING%20THINGS.MD

BOOLEANS / TRUTHY VALUES

- Truthy values should be named in a way that would allow the variable to "sound right" if placed after an "if" statement as if it were parsed by an English speaker. The pattern can be generalized to "Noun_is_Adjective" or "is_Adjective".

```
if subscription_is_paid and
is_above_drinking_age:
    pass
```

HTTPS://GITHUB.COM/LIONHEART/STYLE-GUIDE/BLOB/MASTER/GUIDES/PYTHON/NAMING%20THINGS.MD

BOOLEANS / TRUTHY VALUES

- In addition to "is", there are a few words commonly used to express boolean values. Here's a non-exhaustive list for the most common ones:

Usage	Infix or Prefix
state:	`is_`
ownership:	`has_`
ownership with object:	`contains_`
potential ability:	`should_`
current ability:	`can_`

- Here are a few examples:

```
product_has_description
user_has_delete_permission
should_shave_sheep
```

FUNCTIONS

- Like variables, methods should also be named pretty similarly as if it were a variable representing the same value.

```
class Category(models.Model):
    def is_editable_by_user(self, user):
        pass
    def has_permission(self, permission):
        pass
```

SAMPLE CODE

file: **green_egg_ham_raw.py**

```
times_asked = 0

def ask_if_you_like_green_eggs_and_ham(where="green eggs and ham?"):
    my_question = "do you like " + where
    response = input(my_question)
    return response

def main():
    eggs_are_liked = ask_if_you_like_green_eggs_and_ham()
    times_asked = 1
    questions = {
        "q1": "them here or there?",
        "q2": "them in a house with a mouse?",
        "q3": "them in a box with a fox?",
        "q4": "them on a train?",
        "q5": "them in the dark?",
        "q6": "them on a boat?",
    }
    for i in questions:
        if eggs_are_liked is not True:
            times_asked += 1
            my_question_is = questions[i]
            eggs_are_liked = ask_if_you_like_green_eggs_and_ham(my_question_is)
        else:
            break
    print("Times asked: ", times_asked)
```

SAMPLE CODE

- file: **green_egg_ham.py**

```
def ask_do_you_like_where(where="green eggs and ham?"):
    do_you_like_them = "do you like " + where
    response = input(do_you_like_them)
    return response


def main():
    eggs_are_liked = ask_do_you_like_where()
    num_times_asked = 1
    to_be_asked_dict = {
        "q1": "them here or there?",
        "q2": "them in a house with a mouse?",
        "q3": "them in a box with a fox?",
        "q4": "them on a train?",
        "q5": "them in the dark?",
        "q6": "them on a boat?",
    }
    for each_question in to_be_asked_dict:
        if eggs_are_liked is not True:
            num_times_asked += 1
            question_where = to_be_asked_dict[each_question]
            eggs_are_liked = ask_do_you_like_where(question_where)
        else:
            break
    print("Times asked: ", num_times_asked)
```

THANK YOU

EMAIL: JEREMYTEDWARDS@GMAIL.COM

TWEET: @JEREMYTEDWARDS

GITHUB: HTTPS://GITHUB.COM/JEREMYTEDWARDS/BEGINNER-BOOKS