GDG Seoul

AI 연구자를 위한 클린코드

ML/Data

GDG Seoul

GDG Seoul

#DevFest

Topic

- What is Clean Code
- Use case
- The anti-patterns

한성민, Naver Clova
@kennethan

1

What is Clean Code

깨진 유리창 이론
Broken Windows Theory

만일 한 건물의 유리창이 깨어진 채로 방치 되어 있다면, 곧 **다른 유리창들도 곧 깨어질 것**

여러분의 품질이 떨어지는 코드를 쌓아올리기 시작하는 순간부터
곧 다른 협업자들의 코드 품질도 떨어지기 시작한다는 것

깨진 유리창 이론
What is Clean Code

우리 주변의 깨진 유리창.JPG

DEVFEST 2019

GDG Seoul

```python
def times_table_song(a):
    # TODO: prints the result of all times table printing
    pass

def main():
    # EXPECTED:
    # 2x1 = 2
    # 2x2 = 4
    # ...
    # 2x9 = 18

    times_table_song(2)

if __name__ == '__main__':
    main()
```

구구단을 작성한다고 가정해봅시다.

우리는
TODO로 작성된 코멘트에
내용을 채울 것입니다.

```python
def times_table_song(a):
    for step in range(1, 10):
            result = a * step
            print(str(a) + 'x' + str(step) + ' = ' + str(result))


def main():
    # EXPECTED:
    # 2x1 = 2
    # 2x2 = 4
    # ...
    # 2x9 = 18

    times_table_song(2)


if __name__ == '__main__':
    main()
```

OUTPUT

2x1 = 2
2x2 = 4
…
2x9 = 18

코드의 가독성 문제가 서서히 나타납니다.

이 과정에서, 누군가 코드 품질을
검토하거나, 수정하지 않으면

코드 품질은 점점 악화되기 시작합니다.

```python
def times_table_song(a):
    if a < 2 or a > 9:
        raise ('input number should be larger than 1 or less than 10')
    else:
        for step in range(1, 10):
            result = a * step
            print(str(a) + 'x' + str(step) + ' = ' + str(result))


def main():
    # EXPECTED:
    # 2x1 = 2
    # 2x2 = 4
    # ...
    # 2x9 = 18

    times_table_song(2)


if __name__ == '__main__':
    main()
```

OUTPUT

2x1 = 2
2x2 = 4
…
2x9 = 18

이후 조건 분기가 추가됩니다,

분기(branch)가 늘어나게 되며,

이런 분기가 코드의
복잡성(complexity)을 높이게 됩니다.

```python
def times_table_song(a):
    if a < 2 or a > 9:
        # guard clause
        raise ('input number should be larger than 1 or less than 10')

    for step in range(1, 10):
        # f-string template
        # variable inlining
        print(f'{a}x{step} = {a*step}'


def main():
    # EXPECTED:
    # 2x1 = 2
    # 2x2 = 4
    # ...
    # 2x9 = 18

    times_table_song(2)


if __name__ == '__main__':
    main()
```

OUTPUT

2x1 = 2
2x2 = 4
…
2x9 = 18

개선된 코드입니다.

코드 품질이 크게 저하된 것이 아니기 때문에, 코드 수정에 많은 비용이 들지 않습니다.

지속적인 리뷰로 코드 개선으로 거대한 코드악취로 인한 비용을 줄일 수 있습니다.

보이 스카웃 규칙
The boy scout rule

떠날 때는 찾을 때 보다 캠프장을 더욱 깨끗이 할 것.

**" 여러분의 손을 거치게된 코드는
항상 원래 있었던 상태보다 조금 더 낫게 만들고 떠나라. "**

Robert C. Martin

2

Use Case

코드악취
The code smell

"여러분의 코드에 문제가 발생할 때 생기는 나쁜 냄새"

```python
def introduce():
    message = [
        "안녕하세요 저는 GDG DevFest 2019 발표자에요",
        '취미는 코딩이고, 특기는 똥싸서 남들이 내 똥을 치우게 하는 걸 잘해요!'
    ]

    dt = datetime.now()
    year = dt.year

    if year == 2019:
        for msg in message:
            print(msg)

        if dt.month == 10 and dt.day == 20 and dt.hour == 13:
            print("오 대략 지금이에요!")
    else:
        print("올해는 저의 발표의 순간이 아니군요!")
```

```python
def introduce():
    dt = datetime.now()
    message_by_year = {
        '2019': [
            '안녕하세요 저는 GDG DevFest 2019 발표자에요',
            '취미는 코딩이고, 특기는 똥싸서 남들이 내 똥을 치우게 하는 걸 잘해요!'
        ],
        'default': ['올해는 저의 발표의 순간이 아니군요!']
    }
    message_key = str(dt.year) if str(dt.year) in message_by_year else 'default'
    message = message_by_year[message_key]

    if [dt.year, dt.month, dt.day, dt.hour] == [2019, 10, 20, 13]:
        message.append('오 대략 지금이에요!')

    for msg in message:
        print(msg)
```

너무 많은 분기(Branch)

코드악취 유형 1

동료의 안구를 파괴하고 싶을때만 사용하세요!

```
def my_favorite(weekday, money):
    print('나는요 오늘 뭐가 하고싶냐면요!')

    if weekday == '월요일':
        if money > 5000:
            print('피곤하니깐 스타벅스나 갈레요')

            if starbucks.is_closed():
                print('아 망했어요')
            else:
                print('여기 따듯한 아이스 아메리카노 한잔 주세요')
        else:
            # ...
    # ...
```

```
def my_favorite(weekday, money):
    print('나는요 오늘 뭐가 하고싶냐면요!')
    if weekday in ['토요일', '일요일']:
        print('오늘은 쉬는날!')
        return

    if weekday == '월요일' and money > 5000:
        print('피곤하니깐 스타벅스나 갈레요')
        print('여기 따듯한 아이스 아메리카노 한잔 주세요'
            if not starbucks.is_closed() else '아 망했어요')
        return

    if ...:
        return
```

주석의 남용

코드악취 유형 2

자세하고 친절한 주석을 함수 이름으로 바꿔보세요

```python
def payment(user_id):
    # get user permission information
    # following snippet will get a user information to get user permission from datab
    curs = conn.cursor()
    sql = 'select * from users where user_id=%s limit 1'
    curs.execute(sql, (user_id,))
    row = curs.fetchone()

    if row is None:
        raise(ValueError('User is not exists'))

    permission = row[2]

    # then check the permission
    if permission == ANONYMOUS:
        raise(PermissionError(f'`{permission}` user can not buy without sign-in'))
```

```python
def get_user(user_id):
    curs = conn.cursor()
    sql = f'select * from users where user_id=%s limit 1'
    curs.execute(sql, (user_id,))
    row = curs.fetchone()

    if row is None:
        raise(ValueError('User is not exists'))

    return row


def check_permission(permission):
    if permission == ANONYMOUS:
        raise(PermissionError(f'`{permission}` user can not buy without sign-in'))


def payment(user_id):
    _, _, permission = get_user(user_id)
    check_permission(permission)
```

중복코드

코드악취 유형 3

여러분의 아기자기한 코드를 합쳐보세요!

```python
def get_price_of_meat(meat_type, weight):
    if meat_type == DUCK:
        return (240 * weight) * 100

    if meat_type == CHICKEN:
        return (150 * weight) * 100

    if meat_type == BEEF:
        return (220 * weight) * 100

    if meat_type == PORK:
        return (180 * weight) * 100

    raise NotImplementedError(f'{meat_type} type is not supported yet')
```

```python
def get_price_of_meat(meat_type, weight):
    units = {
        DUCK: 240,
        CHICKEN: 150,
        BEEF: 220,
        PORK: 180
    }

    if meat_type in units.keys():
        unit = units[meat_type]
        return unit * weight * 100

    raise NotImplementedError(f'{meat_type} type is not supported yet')
```

그 밖의 코드악취 케이스

- **너무 긴 메소드**: 메소드 혹은 함수의 내용이 너무 김
- **너무 거대한 클래스**: 클래스가 맡은 역할이 너무 많음
- **상속 거부**: 자식 클래스에서 부모 클래스의 규칙을 무시하고 오버라이드
- **과도한 복잡성**: 간결할 수 있는 코드에 너무 많은 복잡성을 주는 경우
- **게으른 클래스**: 클래스에 부여된 기능이 너무 적은 경우

실제 모델 코드 최적화

코드악취 사례

Attention-based bidirectional LSTM for Classification 구현 중 일부

```
def lstm(vocab, hidden_units, num_layers, max_sequence_length, is_attention, is_bidirectional):
    timesteps = max_sequence_length
    num_classes = 2

    adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.01)

    model = Sequential()
    model.add(Embedding(len(vocab), 100, input_length=35))

    for i in range(num_layers):
        return_sequences = is_attention or (num_layers > 1 and i < num_layers - 1)

        if is_bidirectional:
            model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                         kernel_initializer=initializers.glorot_normal(seed=777),
                                         bias_initializer='zeros')))
        else:
            model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                           kernel_initializer=initializers.glorot_normal(seed=777), bias_initializer=
'zeros'))

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(num_classes, activation='softmax', kernel_initializer=initializers.glorot_normal(
seed=777),
                    bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```


https://github.com/gentaiscool/lstm-attention
arxiv.org/pdf/1805.12307.pdf

실제 모델 코드 최적화

코드악취 사례

1. 가장 먼저 정의했지만

안쓰는 키워드 제거

(Delete unused variable)

```
def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional):
    num_classes = 2

    adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.01)

    model = Sequential()
    model.add(Embedding(len(vocab), 100, input_length=35))

    for i in range(num_layers):
        return_sequences = is_attention or (num_layers > 1 and i < num_layers - 1)

        if is_bidirectional:
            model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                         kernel_initializer=initializers.glorot_normal(seed=777),
                                         bias_initializer='zeros')))
        else:
            model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                           kernel_initializer=initializers.glorot_normal(seed=777), bias_initializer=
'zeros'))

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(num_classes, activation='softmax', kernel_initializer=initializers.glorot_normal(
seed=777),
                    bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```


https://github.com/gentaiscool/lstm-attention

arxiv.org/pdf/1805.12307.pdf

실제 모델 코드 최적화

코드악취 사례

2. 직접 기입되어 있는

설정 인수들을 전역 변수로 추출

```
NUM_CLASSES = 2
LEARNING_RATE = 0.0005
BETA_1_PARAM = 0.9
BETA_2_PARAM = 0.999
EPSILON = 1e-08
DECAY_RATE = 0.01

EMBEDDING_OUTPUT_DIM = 100
EMBEDDING_INPUT_LEN = 35

def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional):
    adam = optimizers.Adam(
        lr=LEARNING_RATE, beta_1=BETA_1_PARAM, beta_2=BETA_2_PARAM, epsilon=EPSILON, decay=DECAY_RATE)
    model = Sequential()
    model.add(Embedding(len(vocab), EMBEDDING_OUTPUT_DIM, input_length=EMBEDDING_INPUT_LEN))

    for i in range(num_layers):
        return_sequences = is_attention or (num_layers > 1 and i < num_layers - 1)

        if is_bidirectional:
            model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                         kernel_initializer=initializers.glorot_normal(seed=777),
                                         bias_initializer='zeros')))
        else:
            model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                           kernel_initializer=initializers.glorot_normal(seed=777), bias_initializer='zer
os'))

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=initializers.glorot_normal(seed
=777),
                    bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```

DEVFEST 2019

GDG Seoul

https://github.com/gentaiscool/lstm-attention

arxiv.org/pdf/1805.12307.pdf

실제 모델 코드 최적화

코드악취 사례

3. 복잡성이 높은 로직

함수 추출

```
NUM_CLASSES = 2
LEARNING_RATE = 0.0005
BETA_1_PARAM = 0.9
BETA_2_PARAM = 0.999
EPSILON = 1e-08
DECAY_RATE = 0.01

EMBEDDING_OUTPUT_DIM = 100
EMBEDDING_INPUT_LEN = 35

def add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional):
    if is_bidirectional:
        model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                     kernel_initializer=initializers.glorot_normal(seed=777),
                                     bias_initializer='zeros')))
        return

    model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                   kernel_initializer=initializers.glorot_normal(seed=777), bias_initializer='zeros'))


def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional):
    adam = optimizers.Adam(
        lr=LEARNING_RATE, beta_1=BETA_1_PARAM, beta_2=BETA_2_PARAM, epsilon=EPSILON, decay=DECAY_RATE)
    model = Sequential()
    model.add(Embedding(len(vocab), EMBEDDING_OUTPUT_DIM, input_length=EMBEDDING_INPUT_LEN))

    for i in range(num_layers):
        return_sequences = is_attention or (num_layers > 1 and i < num_layers - 1)
        add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional)

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=initializers.glorot_normal(seed=777),
                    bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```

DEVFEST 2019

GDG Seoul

https://github.com/gentaiscool/lstm-attention
arxiv.org/pdf/1805.12307.pdf

실제 모델 코드 최적화

코드악취 사례

4. 조건 식 단순화,

중복 코드 함수 추출

```
NUM_CLASSES = 2
LEARNING_RATE = 0.0005
BETA_1_PARAM = 0.9
BETA_2_PARAM = 0.999
EPSILON = 1e-08
DECAY_RATE = 0.01

EMBEDDING_OUTPUT_DIM = 100
EMBEDDING_INPUT_LEN = 35
GLOROT_NORMAL_SEED = 777

def get_initializer():
    return initializers.glorot_normal(seed=GLOROT_NORMAL_SEED)


def add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional):
    if is_bidirectional:
        model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                     kernel_initializer=get_initializer(), bias_initializer='zeros')))
        return

    model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                   kernel_initializer=get_initializer(), bias_initializer='zeros'))


def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional):
    adam = optimizers.Adam(
        lr=LEARNING_RATE, beta_1=BETA_1_PARAM, beta_2=BETA_2_PARAM, epsilon=EPSILON, decay=DECAY_RATE)
    model = Sequential()
    model.add(Embedding(len(vocab), EMBEDDING_OUTPUT_DIM, input_length=EMBEDDING_INPUT_LEN))

    for i in range(num_layers):
        return_sequences = is_attention or min(1, i+1) < num_layers
        add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional)

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=get_initializer(), bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```


https://github.com/gentaiscool/lstm-attention

arxiv.org/pdf/1805.12307.pdf

실제 모델 코드 최적화

코드악취 사례

5. 모델 혹은 Oprimizer의 인수 정보들을 전역 변수에서 설정 프레임워크를 이용한 파라미터로 전환

*** google/gin-config**

```
def add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional):
    if is_bidirectional:
        model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                     kernel_initializer=get_initializer(), bias_initializer='zeros')))
        return

    model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                   kernel_initializer=get_initializer(), bias_initializer='zeros'))

@gin.configurable
def get_initializer(glorot_normal_seed=777):
    return initializers.glorot_normal(seed=glorot_normal_seed)


@gin.configurable
def get_adam_optimizer(learning_rate, beta_1, beta_2, epsilon, decay_rate):
    return optimizers.Adam(lr=learning_rate, beta_1=beta_1, beta_2=beta_2, epsilon=epsilon, decay=decay_rate)


@gin.configurable
def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional,
         embedding_output_dim=35, embedding_input_len=10):
    model = Sequential()
    model.add(Embedding(len(vocab), embedding_output_dim,, input_length=embedding_input_len)

    for i in range(num_layers):
        return_sequences = is_attention or min(1, i+1) < num_layers
        add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional)

        if not is_attention:
            continue

        model.add(AttentionWithContext())
        model.add(Addition())

    adam = get_get_adam_optimizer()
    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=get_initializer(), bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```


https://github.com/gentaiscool/lstm-attention
arxiv.org/pdf/1805.12307.pdf

실제 모델 코드 최적화

코드악취 사례

6. 코드 컨벤션 공통화

=> ', " 두가지로 사용하던 따옴표를 '로 통일

=> 함수 사이를 2줄 넘김으로 구분

=> 1줄을 넘어서는 함수 인수의 intent 재조정

```
def add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional):
    if is_bidirectional:
        model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                     kernel_initializer=get_initializer(), bias_initializer='zeros')))
        return

    model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                   kernel_initializer=get_initializer(), bias_initializer='zeros'))


@gin.configurable
def get_initializer(glorot_normal_seed=777):
    return initializers.glorot_normal(seed=glorot_normal_seed)


@gin.configurable
def get_adam_optimizer(learning_rate, beta_1, beta_2, epsilon, decay_rate):
    return optimizers.Adam(lr=learning_rate, beta_1=beta_1, beta_2=beta_2, epsilon=epsilon, decay=decay_rate)


@gin.configurable
def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional,
         embedding_output_dim=35, embedding_input_len=10):
    model = Sequential()
    model.add(Embedding(len(vocab), embedding_output_dim,, input_length=embedding_input_len)

    for i in range(num_layers):
        return_sequences = is_attention or min(1, i + 1) < num_layers
        add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional)

        if not is_attention:
            continue

        model.add(AttentionWithContext())
        model.add(Addition())

    adam = get_get_adam_optimizer()
    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=get_initializer(), bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    model.summary()

    return model
```

DEVFEST 2019

GDG Seoul

https://github.com/gentaiscool/lstm-attention

arxiv.org/pdf/1805.12307.pdf

```python
def lstm(vocab, hidden_units, num_layers, max_sequence_length, is_attention, is_bidirectional):
    timesteps = max_sequence_length
    num_classes = 2

    adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.01)

    model = Sequential()
    model.add(Embedding(len(vocab), 100, input_length=35))

    for i in range(num_layers):
        return_sequences = is_attention or (num_layers > 1 and i < num_layers - 1)

        if is_bidirectional:
            model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                         kernel_initializer=initializers.glorot_normal(seed=777),
                                         bias_initializer='zeros')))
        else:
            model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                           kernel_initializer=initializers.glorot_normal(seed=777), bias_initializer='zeros'))

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(num_classes, activation='softmax', kernel_initializer=initializers.glorot_normal(seed=777),
                    bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```

```python
def add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional):
    if is_bidirectional:
        model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                     kernel_initializer=get_initializer(), bias_initializer='zeros')))
        return

    model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                   kernel_initializer=get_initializer(), bias_initializer='zeros'))


@gin.configurable
def get_initializer(glorot_normal_seed=777):
    return initializers.glorot_normal(seed=glorot_normal_seed)


@gin.configurable
def get_adam_optimizer(learning_rate, beta_1, beta_2, epsilon, decay_rate):
    return optimizers.Adam(lr=learning_rate, beta_1=beta_1, beta_2=beta_2, epsilon=epsilon, decay=decay_rate)


@gin.configurable
def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional,
         embedding_output_dim=35, embedding_input_len=10):
    model = Sequential()
    model.add(Embedding(len(vocab), embedding_output_dim,, input_length=embedding_input_len)

    for i in range(num_layers):
        return_sequences = is_attention or min(1, i + 1) < num_layers
        add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional)

        if not is_attention:
            continue

        model.add(AttentionWithContext())
        model.add(Addition())

    adam = get_get_adam_optimizer()
    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=get_initializer(),
        bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    model.summary()

    return model
```

DEVFEST 2019

GDG Seoul

https://github.com/gentaiscool/lstm-attention
arxiv.org/pdf/1805.12307.pdf

```python
def lstm(vocab, hidden_units, num_layers, max_sequence_length, is_attention, is_bidirectional):
    timesteps = max_sequence_length
    num_classes = 2

    adam = optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.01)

    model = Sequential()
    model.add(Embedding(len(vocab), 100, input_length=35))

    for i in range(num_layers):
        return_sequences = is_attention or (num_layers > 1 and i < num_layers - 1)

        if is_bidirectional:
            model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                         kernel_initializer=initializers.glorot_normal(seed=777),
                                         bias_initializer='zeros')))
        else:
            model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                           kernel_initializer=initializers.glorot_normal(seed=777), bias_initializer='zeros'))

        if is_attention:
            model.add(AttentionWithContext())
            model.add(Addition())

    model.add(Dense(num_classes, activation='softmax', kernel_initializer=initializers.glorot_normal(seed=777),
                    bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
    model.summary()

    return model
```

실제 작업 영역

```python
def add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional):
    if is_bidirectional:
        model.add(Bidirectional(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                                     kernel_initializer=get_initializer(), bias_initializer='zeros')))
        return

    model.add(LSTM(hidden_units, return_sequences=return_sequences, dropout=0.2,
                   kernel_initializer=get_initializer(), bias_initializer='zeros'))


@gin.configurable
def get_initializer(glorot_normal_seed=777):
    return initializers.glorot_normal(seed=glorot_normal_seed)


@gin.configurable
def get_adam_optimizer(learning_rate, beta_1, beta_2, epsilon, decay_rate):
    return optimizers.Adam(lr=learning_rate, beta_1=beta_1, beta_2=beta_2, epsilon=epsilon, decay=decay_rate)


@gin.configurable
def lstm(vocab, hidden_units, num_layers, is_attention, is_bidirectional,
         embedding_output_dim=35, embedding_input_len=10):
    model = Sequential()
    model.add(Embedding(len(vocab), embedding_output_dim,, input_length=embedding_input_len)

    for i in range(num_layers):
        return_sequences = is_attention or min(1, i + 1) < num_layers
        add_lstm_layer(model, hidden_units, return_sequences, is_bidirectional)

        if not is_attention:
            continue

        model.add(AttentionWithContext())
        model.add(Addition())

    adam = get_get_adam_optimizer()
    model.add(Dense(NUM_CLASSES, activation='softmax', kernel_initializer=get_initializer(),
        bias_initializer='zeros'))
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    model.summary()

    return model
```

DEVFEST 2019

GDG Seoul

https://github.com/gentaiscool/lstm-attention
arxiv.org/pdf/1805.12307.pdf

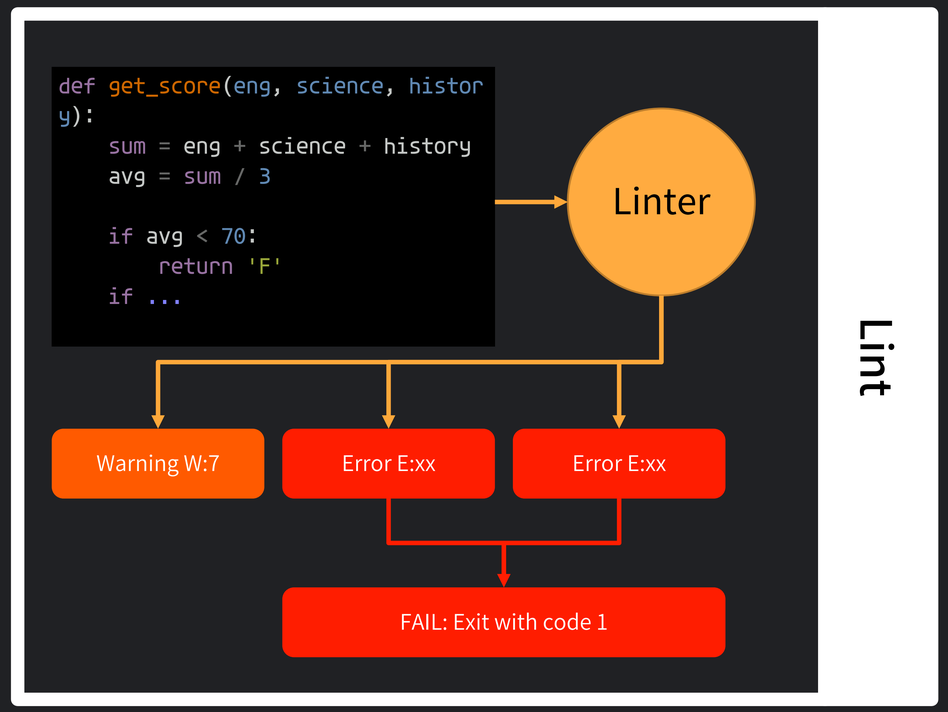
github.com/IDSIA/sacred

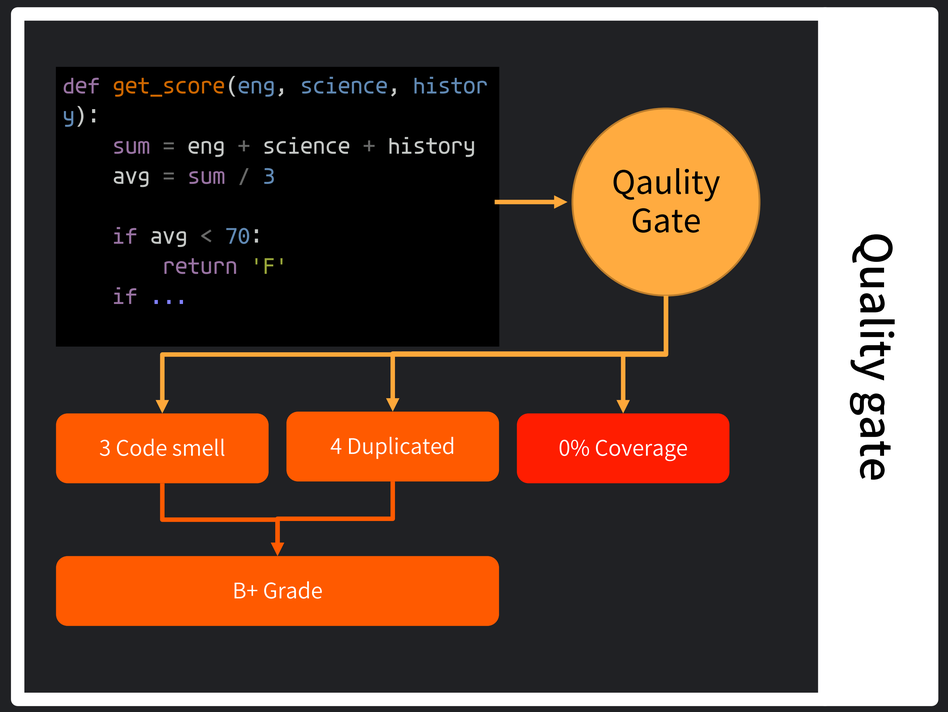
Configuration Manager

```python
from numpy.random import permutation
from sklearn import svm, datasets

C = 1.0
gamma = 0.7

iris = datasets.load_iris()
perm = permutation(iris.target.size)
iris.data = iris.data[perm]
iris.target = iris.target[perm]
clf = svm.SVC(C, 'rbf', gamma=gamma)
clf.fit(iris.data[:90],
        iris.target[:90])
print(clf.score(iris.data[90:],
                iris.target[90:]))
```

```python
from numpy.random import permutation
from sklearn import svm, datasets
from sacred import Experiment
ex = Experiment('iris_rbf_svm')

@ex.config
def cfg():
  C = 1.0
  gamma = 0.7

@ex.automain
def run(C, gamma):
  iris = datasets.load_iris()
  per = permutation(iris.target.size)
  iris.data = iris.data[per]
  iris.target = iris.target[per]
  clf = svm.SVC(C, 'rbf', gamma=gamma)
  clf.fit(iris.data[:90],
          iris.target[:90])
  return clf.score(iris.data[90:],
                   iris.target[90:])
```

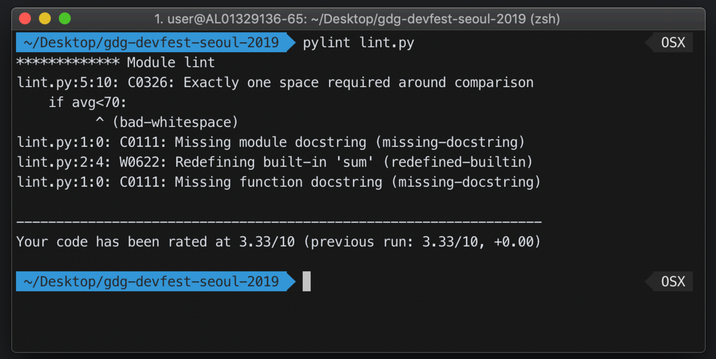
지속적인 코드 관리


def get_score(eng, science, histor
y):
sum = eng + science + history
avg = sum / 3
if avg < 70:
return 'F'
if ...
Linter
Warning W:7
Error E:xx
Error E:xx
FAIL: Exit with code 1
Lint

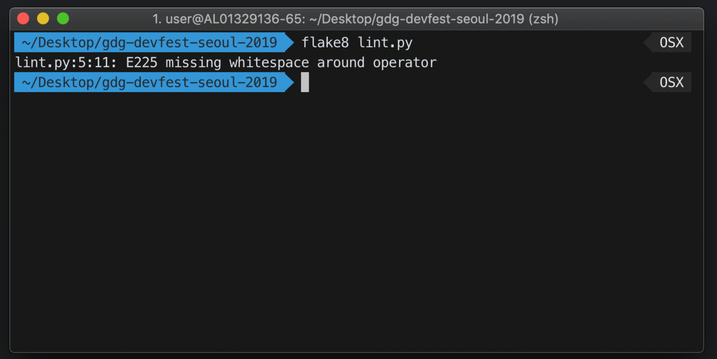

def get_score(eng, science, histor
y):
sum = eng + science + history
avg = sum / 3
if avg < 70:
return 'F'
if ...
Qaulity Gate
3 Code smell
4 Duplicated
0% Coverage
B+ Grade
Quality gate

파이썬 Lint 도구

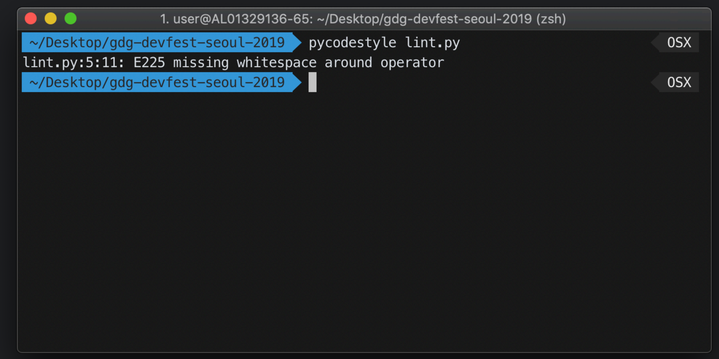
pylint



flake8

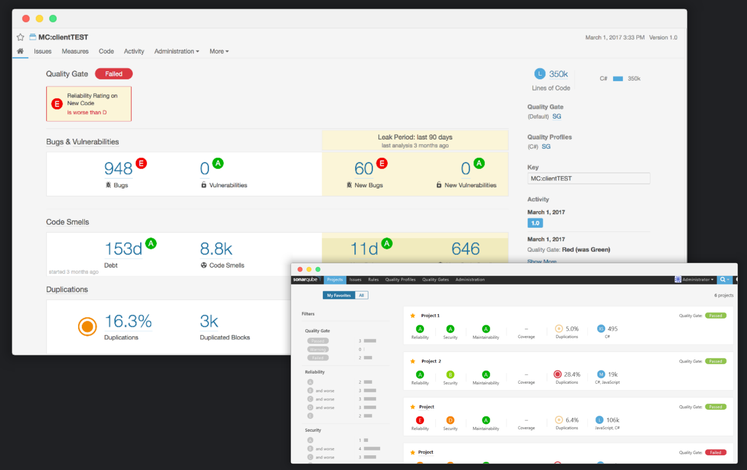


pycodestyle

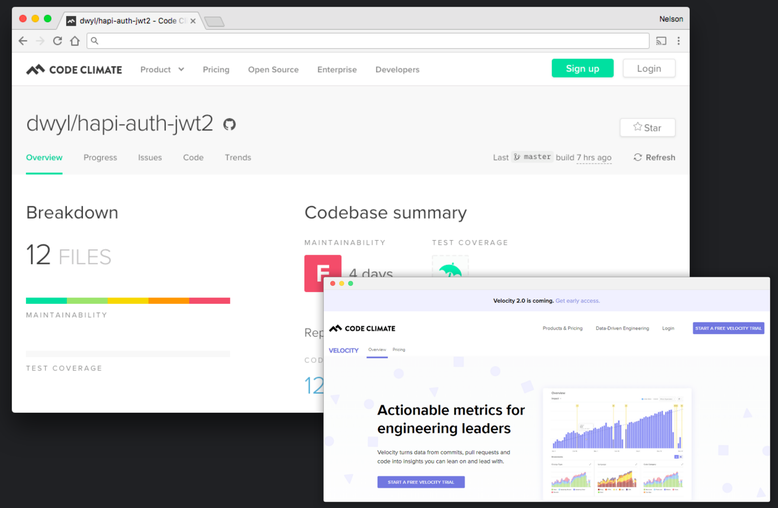


Quality gate 도구

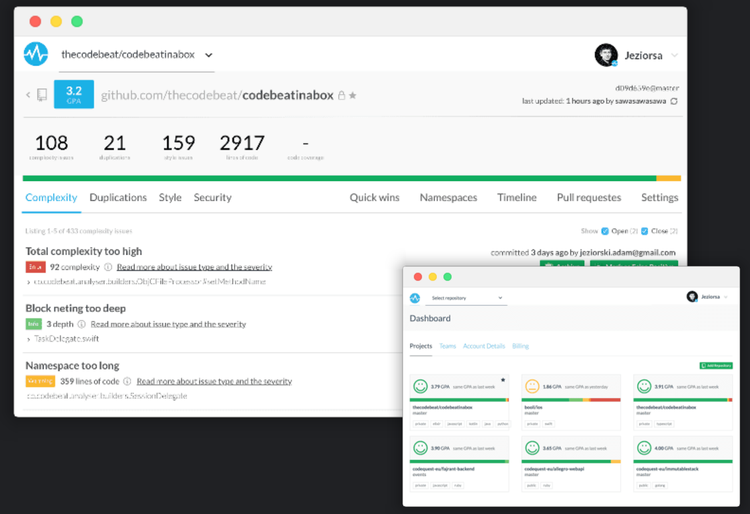
Sonarqube



codeclimate



codebeat



DEVFEST 2019

GDG Seoul

3

The anti-patterns

If/Switch 절의 남용

Anti patterns #1

```
def init_data(name, mode='train'):
    if mode == 'train':
        test_data = load_test_data('d_1.tsv')
        if name == 'click':
            test_data = load_test_data('d_click_2019.tsv')
        elif name == 'conversion':
            test_data = load_test_data('d_click_2019_with_conversion.tsv')
        elif name == ''
            test_data = load_test_data('d_click_2019_with_drop')
    else:
        if name == 'click':
            test_data = load_test_data('train_click_2018.tsv')
        elif name == 'conversion':
            test_data = load_test_data('train_d_click_2019_with_conversion.tsv')
        elif name == ''
            test_data = load_test_data('train_d_click_2019_with_drop')

# ...
```

Key access를 통한 분기대체

Anti patterns #1

```
def get_item(condition):
    if condition == 'a':
        return 'apple'
    elif condition == 'b':
        return 'banana'
    else:
        return 'grape'
```

```
def get_item(condition):
    conditions = {
        'a': 'apple',
        'b': 'banana',
        'otherwise': 'grape'
    }

    if condition in conditions.keys():
        return conditions[condition]

    return conditions['otherwise']
```

If/Switch 절의 대체

Anti patterns #1

```python
def init_data(name, mode='train'):
    if mode == 'train':
        test_data = load_test_data('d_1.tsv')
        if name == 'click':
            test_data = load_test_data('d_click_2019.tsv')
        elif name == 'conversion':
            test_data = load_test_data('d_click_2019_with_conversion.tsv')
        elif name == ''
            test_data = load_test_data('d_click_2019_with_drop')
    else:
        if name == 'click':
            test_data = load_test_data('train_click_2018.tsv')
        elif name == 'conversion':
            test_data = load_test_data('train_d_click_2019_with_conversion.tsv')
        elif name == ''
            test_data = load_test_data('train_d_click_2019_with_drop')

# ...
```

```python
def init_data(name, mode='train'):
    data_path = {
        'train': {
            'click': 'd_click_2019.tsv',
            'conversion': 'd_click_2019_with_conversion.tsv',
            '': 'd_click_2019_with_drop'
        },
        'test': {
            'click': 'train_click_2018.tsv',
            'conversion': 'train_d_click_2019_with_conversion.tsv',
            '': 'train_d_click_2019_with_drop'
        }
    }

    if not mode in data_path.keys():
        raise ValueError(f'`mode` is not supported type')

    if not name in data_path[mode].keys():
        raise ValueError(f'`mode`.`name` is no exists data')

    return data_path[mode][name]
```

데이터 초기화와 데이터 삽입 로직의 분리

Anti patterns #2

```
def pre_process_data(data):
    output_data = []
    for row in data:
        output_Data.append(get_name_of_label(
            row['label']), row['value'])

    return output_data
```

고차함수를 이용한 대체

Anti patterns #2

```python
def get_data(list_data):
    result = []
    for item in list_data:
        result.append(item['key'])
    return result
```

```python
def get_data(list_data):
    return map(
        lambda item: item['key'], list_data)
```

데이터 초기화와 데이터 삽입 로직 병합

Anti patterns #2

```
def pre_process_data(data):
    output_data = []
    for row in data:
        output_Data.append(
        get_name_of_label(row['label']), row['value'])

    return output_data
```

```
def pre_process_data(data):
    return list(map(lambda row:
        get_name_of_label(row['label']), data))
```

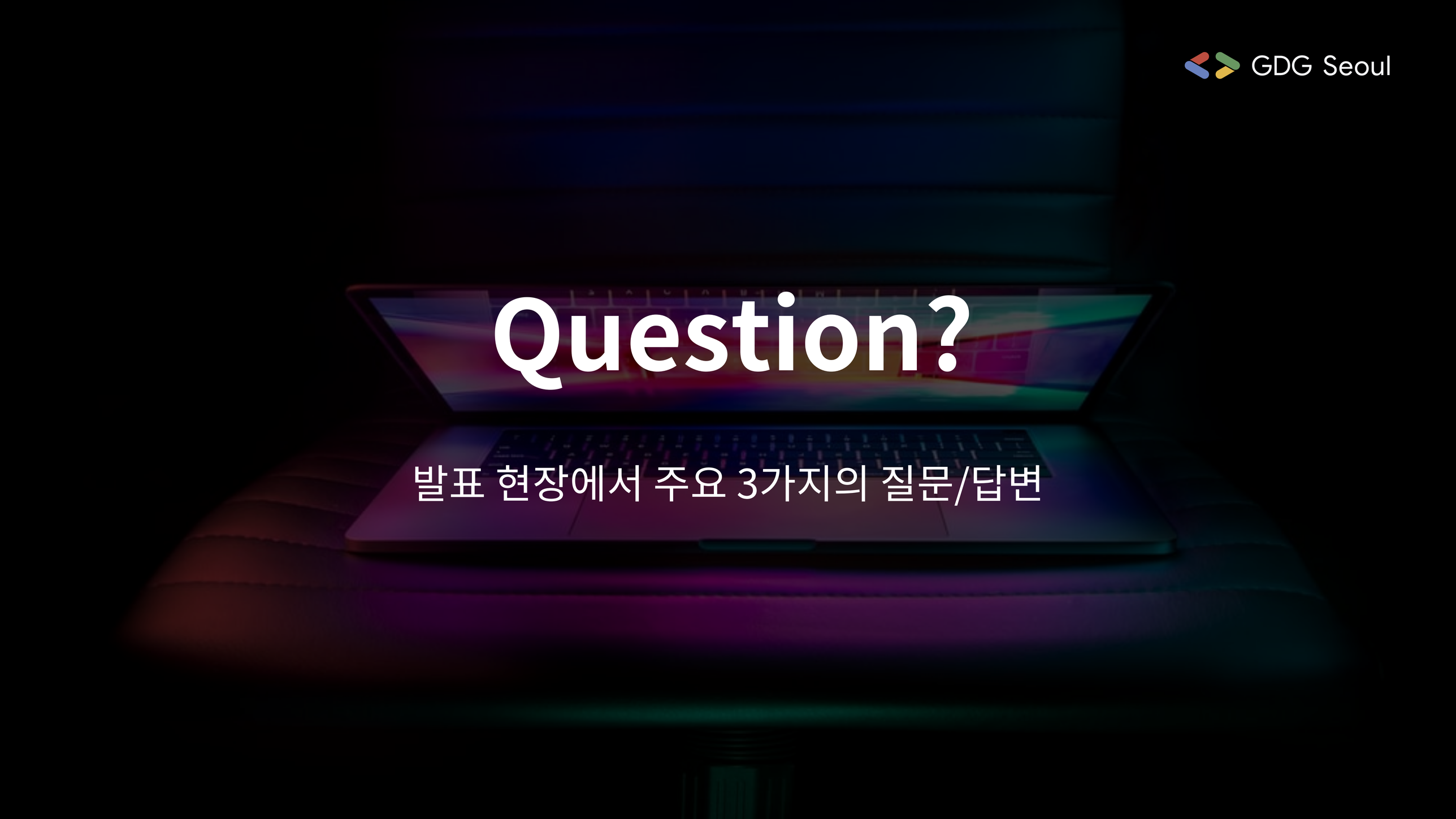
타입의 부재

Anti patterns #3

```
def get_test_data():
    data = load_data('dev_env_click_3000.csv')
    return pre_process(data)

@mytest.automain
def get_train(options):
    model = build_model(options)
    model.train(get_test_data())
```

```
def get_test_data() -
> List[List[Dict[str, str, str, int, int, int]]]:
    data = load_data('dev_env_click_3000.csv')
    return pre_process(data)

@mytest.automain
def get_train(options: Options) -> None:
    model = build_model(options)
    model.train(get_test_data())
```

GDG Seoul



Question?

발표 현장에서 주요 3가지의 질문/답변

데이터와 로직이 밀접하게 붙어있는 경우 어떻게 분리해야 할까요?

Question #1

데이터가 데이터베이스와 같은 지속레이어(Persistent layer)에 있을 경우 Entity로서
데이터가 유효할 것이라는 확신을 가지고, 코드를 관리하는 방법이 있습니다.

반면에 API나, Auth, Payment같은 유효성을 보장할 수 없는 경우에는
DTO나 Service 레벨에서 NullObject와 같은 방법을 통해 데이터 유효성을 보장해주는 방법이
코드 품질을 높일 수 있게 됩니다.

일정의 압박이 있을 때도 클린코드의 규칙을 항상 준수하시나요?

Question #2

협업을 위해서 Git과 같은 도구를 사용하는 경우,
일정량의 코드 품질이나 테스트를 통과하지 못하면 CI(Continous Integration)에서 Fail 처리를 하고
이 경우 GitHub UI에서 Merge 버튼 자체를 비활성화 시키게 됩니다.

Protected Branch를 통해, 일정이 바쁘더라도 Feature에 대해서는 코드 품질을 적정량 준수시켜야 하는 것이죠,

반면 Hotfix와 같은 긴급 오류 수정에 대해서는 CI로 나타나는 오류나, 코드리뷰를 거치지 않더라도 Merge가 가능하도록 열어두었습니다, 다만 코드 리뷰를 거치지 않기 때문에 코드 자체에 책임을 고스란히 본인이 지게 된다는 의미가 있지요.

주석을 어느 수준까지 달아야 클린코드라 할 수 있을까요?

Question #3

이것은 업무의 상황에 따라 다르게 느껴집니다.
제 주관으로는 업무 도메인 지식이 얼마나 로직에 녹아있는지에 따라 다를 것 같습니다.

예를들어 인증로직이나, 유틸리티 로직의 경우 로직이 복잡하더라도, 협업에 있어 함수만 잘 나눠주면 이해하는데 문제가 없기 때문에 주석을 얼마든지 함수로 추출할 수 있습니다.

다만 업무자체에 이해가 필요한, 도메인 로직들, 예를들어 대화에서 의문형 대화만을 찾을 때 쓰이는 평가방식이 세분화 되는 로직을 짜야할 경우 함수 이름만으로 나타내기 어려울 것입니다.

또한 함수 자체에 구체적인 설명이 필요한 경우에서 DocString으로 함수에 대한 주석을 달아주는 것을 권장합니다.

GDG Seoul

Thank you!

https://github.com/KennethanCeyer

https://facebook.com/han.sungmin