2022年11月1日(火)
3,4,5限 ＠研A302

プロジェクト演習 テーマD

第5回

担当：CS学部 講師 伏見卓恭
連絡先：fushimity@edu.teu.ac.jp

授業の流れ

- 第 1回:実験環境の構築／Python，Gitの復習／CUIゲームの開発
- 第 2回:Tkinterの基礎
- 第 3回:TkinterによるGUIゲーム開発
- 第 4回:PyGameの基礎
- 第 5回:PyGameによるゲーム開発
- 第 6回:ゲーム開発演習
- 第 7回:成果発表

本日のお品書き

1. サンプルゲームで遊んでみる
2. コード規約
3. オブジェクト指向プログラミングによる可読性，拡張性の向上
4. Pygameを使ってゲームの開発（後半）

3限：Pygameの応用

mainブランチになっていることを確認する

ProjExD2022/ex05/フォルダを作成する

pygameに関する情報の確認

- pipコマンドで確認する

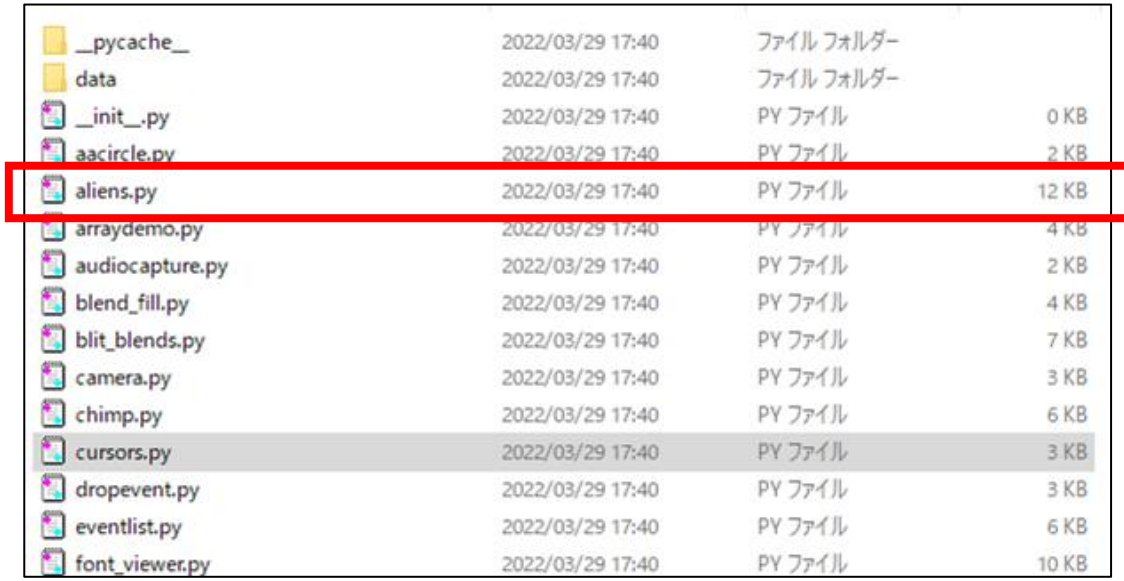
```
(ProjExD) C:¥Users¥fsmtkys>pip show pygame
Name: pygame
Version: 2.1.2
Summary: Python Game Development
Home-page: https://www.pygame.org
Author: A community project.
Author-email: pygame@pygame.org
License: LGPL
Location: c:¥users¥fsmtkys¥appdata¥roaming¥python¥python37¥site-packages
Requires:
Required-by:
```

←大文字のフォルダ名も小文字で表示されている

- Locationのパスをコピーしてエクスプローラーのアドレスバーに貼り付けて移動
  - pygameフォルダ内にpygameモジュールのプログラムがある

サンプルゲームで遊んでみよう

- pygameモジュール内のexamplesにあるサンプルゲームを実行する

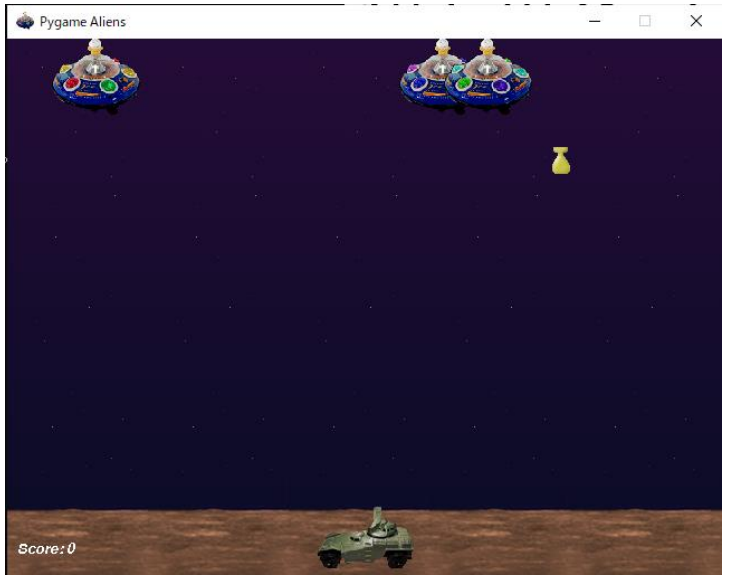


←aliens.pyを実行する(音量注意)

```
(ProjExD) C:¥Users¥fsmtkys>python -m pygame.examples.aliens
```

↑

「python -m モジュール名」でモジュールのプログラムを実行できる



aliens.py（1/4）：import，定数定義部

```
26   import random
27   import os
28
29   # import basic pygame modules
30   import pygame as pg
31
32   # see if we can load more than standard BMP
33   if not pg.image.get_extended():
34       raise SystemExit("Sorry, extended image module required")
35
36
37   # game constants
38   MAX_SHOTS = 2  # most player bullets onscreen
39   ALIEN_ODDS = 22  # chances a new alien appears
40   BOMB_ODDS = 60  # chances a new bomb will drop
41   ALIEN_RELOAD = 12  # frames between new aliens
42   SCREENRECT = pg.Rect(0, 0, 640, 480)
43   SCORE = 0
44
45   main_dir = os.path.split(os.path.abspath(__file__))[0]
46
47
48 > def load_image(file):
56
57
58 > def load_sound(file):
```

モジュールのimport

定数の定義
※定数は全部大文字が基本

aliens.py（2/4）：クラス定義部分

```
 79 > class Player(pg.sprite.Sprite):
109
110
111 > class Alien(pg.sprite.Sprite):
135
136
137 > class Explosion(pg.sprite.Sprite):
162
163
164 > class Shot(pg.sprite.Sprite):
183
184
185 > class Bomb(pg.sprite.Sprite):
209
210
211 > class Score(pg.sprite.Sprite):
229
230
231 > def main(winstyle=0):
396
397
398   # call the "main" function if running this script
399   if __name__ == "__main__":
400       main()
401       pg.quit()
```

Spriteクラスを拡張した
独自クラスを定義

Group()コンストラクタにより，
Spriteクラスのサブクラスのインスタンスを
複数扱うことができる
↓

```
278     # Initialize Game Groups
279     aliens = pg.sprite.Group()
280     shots = pg.sprite.Group()
281     bombs = pg.sprite.Group()
282     all = pg.sprite.RenderUpdates()
283     lastalien = pg.sprite.GroupSingle()
```

importではなく
コマンドラインから実行された際に動く処理

aliens.py（3/4）：Alienクラス

```
111  class Alien(pg.sprite.Sprite):
112      """An alien space ship. That slowly moves down the screen."""
113
114      speed = 13
115      animcycle = 12
116      images = []
117
118      def __init__(self):
119          pg.sprite.Sprite.__init__(self, self.containers)
120          self.image = self.images[0]
121          self.rect = self.image.get_rect()
122          self.facing = random.choice((-1, 1)) * Alien.speed
123          self.frame = 0
124          if self.facing < 0:
125              self.rect.right = SCREENRECT.right
126
127      def update(self):
128          self.rect.move_ip(self.facing, 0)
129          if not SCREENRECT.contains(self.rect):
130              self.facing = -self.facing
131              self.rect.top = self.rect.bottom + 1
132              self.rect = self.rect.clamp(SCREENRECT)
133          self.frame = self.frame + 1
134          self.image = self.images[self.frame // self.animcycle % 3]
```

aliens.py（4/4）：main関数の一部

```
231    def main(winstyle=0):
232        # Initialize pygame
233        if pg.get_sdl_version()[0] == 2:
234            pg.mixer.pre_init(44100, 32, 2, 1024)
235        pg.init()
236        if pg.mixer and not pg.mixer.get_init():
237            print("Warning, no sound")
238            pg.mixer = None
```

```
306        while player.alive():
307
308            # get input
309            for event in pg.event.get():
310                if event.type == pg.QUIT:
311                    return
312                if event.type == pg.KEYDOWN and event.key == pg.K_ESCAPE:
313                    return
314                elif event.type == pg.KEYDOWN:
```

ProjExD2022/ex05にコピー

- aliens.pyとdataフォルダをex05の下にコピーする

- 定数の値を変更して，挙動の違いを確認してみよう(P.7 38~41行目)
- コメントを見て定数が何を定義しているのか考えてみよう
  - MAX_SHOTは画面上に表示される自分の弾の数を設定している
  - ALIEN_ODDは敵が生成される確率を定義している
  - 他の定数についても試すか考えてみよう

練習問題：ex05/dodge_bomb.py

前回実装したex04/dodge_bomb.pyにクラスを導入し，
可読性，拡張性を高める

たとえば，

- Birdクラス
  - イニシャライザ：画像load，拡大縮小，Rect取得，座標設定
  - インスタンス変数：Surface型sfc，Rect型rct
  - クラス変数：キー種と速度の対応表（辞書）key_delta
  - インスタンスメソッド：座標を移動させるupdate()
- Bombクラス
  - イニシャライザ：Surface作成，色・大きさ設定，Rect取得，座標設定
  - インスタンス変数：Surface型sfc，Rect型rct
  - インスタンスメソッド：座標を移動させるupdate()

コード規約　本日課題のチェック項目

- コード規約：https://pep8-ja.readthedocs.io/ja/latest/
  - インデント，行中改行
  - 1行の長さ
  - 演算子の位置
  - 空行の数
  - importの書き方
  - 余分な空白は入れない
  - 命名規則
  - 条件文の書き方
- わかりやすい記事：https://qiita.com/simonritchie/items/bb06a7521ae6560738a7
- 「一貫性にこだわりすぎるのは、狭い心の現れである」
  つまり，規約に囚われすぎない臨機応変さも必要

練習問題：ex05/dodge_bomb.py

1. Screenクラスを作成せよ
   - __init__(タイトル，幅・高さタプル，背景画像ファイル名)
     - タイトルを設定する
     - 幅・高さタプルを指定して，スクリーン用のSurfaceを生成する
     - スクリーン用のSurfaceのRectを取得する
     - 背景画像用のSurfaceを生成し，Rectを取得する
   - インスタンス変数
     - Surface sfc：スクリーン用のSurface
     - Rect rct：スクリーン用のRect
     - Surface bgi_sfc：背景画像用のSurface
     - Rect bgi_rct：背景画像用のRect
   - インスタンスメソッド：blit()
     - Surfaceクラスのblitメソッドを呼び出す
- main関数内の前回練習1の部分をScreenクラスを使うように改めよ

練習問題：ex05/dodge_bomb.py

2. Birdクラスを作成せよ
   - __init__(画像ファイル名，拡大率，初期座標タプル)
     - 画像ファイルを読み込み，画像用のSurfaceを生成する
     - 画像を拡大する
     - 画像用のSurfaceのRectを取得する
     - Rectに座標を設定する
   - インスタンス変数
     - Surface sfc：画像用のSurface
     - Rect rct：画像用のRect
   - クラス変数
     - dict key_delta：押されたキーの種類と移動速度の対応表

練習問題：ex05/dodge_bomb.py

2. Birdクラスを作成せよ
   - インスタンスメソッド：blit(Screenオブジェクト)
     - Screenオブジェクトのインスタンス変数であるsfc（Surfaceクラスのインスタンス）のblitメソッドを呼び出す
   - インスタンスメソッド：update(Screenオブジェクト)
     - get_pressedを用いて，キーの押下状態を取得する
     - クラス変数key_deltaを用いて，押下状態に応じてこうかとんを移動する
     - check_boundを用いて，こうかとんの位置がScreenに収まっているかチェックする
     - 必要に応じて，こうかとんの位置を移動前に戻す
     - 上記のblitメソッドを呼び出す

- main関数内の前回練習3,4,8の部分をBirdクラスを使うように改めよ

練習問題：ex05/dodge_bomb.py

3．Bombクラスを作成せよ

- __init__(色タプル，半径，速度タプル，Screenオブジェクト)
  - 爆弾用のSurfaceを生成する
  - 色タプル，半径を元にSurfaceに円を描画する
  - 爆弾用のSurfaceのRectを取得する
  - Rectに乱数で座標を設定する（乱数の範囲はScreenオブジェクトのインスタンス変数であるrctから取得する）

- インスタンス変数
  - Surface sfc：爆弾用のSurface
  - Rect rct：爆弾用のRect
  - int vx, vy：速度

練習問題：ex05/dodge_bomb.py

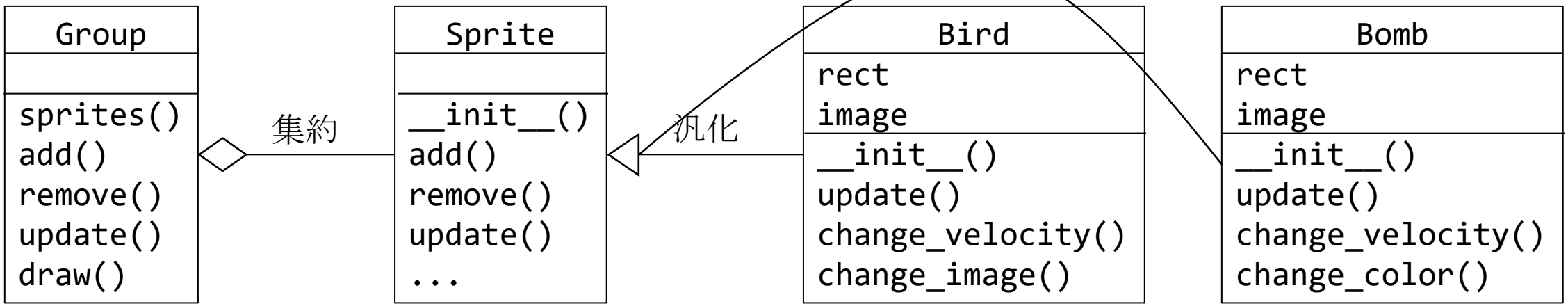
3. Bombクラスを作成せよ
   - インスタンスメソッド：blit(Screenオブジェクト)
     - Screenオブジェクトのインスタンス変数であるsfc（Surfaceクラスのインスタンス）のblitメソッドを呼び出す
   - インスタンスメソッド：update(Screenオブジェクト)
     - move_ipを用いて，速度に応じて爆弾を移動する
     - check_boundを用いて，爆弾の位置がScreenに収まっているかチェックする
     - 必要に応じで，速度vx,vyを更新する
     - 上記のblitメソッドを呼び出す

- main関数内の前回練習5,6,8の部分をBombクラスを使うように改めよ

※コミットコメントを「3限基本機能実装完了」としよう
そしてリモートリポジトリにプッシュしよう

【Advanced】スプライト処理

- 1つの画面の絵を，構成する複数の要素に分解し，それらを画面上の任意の位置で合成して表示する技術のこと
- ゲームのような，構成要素の位置が変動する場合に適した画像処理法
- Pygameでは，spriteモジュールが用意されている
- spriteモジュールには，SpriteクラスとGroupクラスがある
- Spriteクラスを拡張したオリジナルクラス(BirdやBomb)を作成し，そのインスタンス群を保持するコンテナとしてGroupクラスを用いる



【Advanced】練習問題

4. BirdクラスとBombクラスを，Spriteクラスのサブクラスとして定義せよ
5. 次の①②③④のうち，未対応のものを記述せよ

特定の画像をゲーム画面に表示するためのシンプルな基底クラスです。
このクラスを継承する場合は、① Sprite.update命令をオーバーライドして、
② Sprite.image属性と③ Sprite.rect属性を設定してください。
コンストラクタ実行時には望む数のGroupインスタンスを引数として設定することができ、
所属Groupとして追加されます。

このSpriteクラスを継承して使う場合は、
Groupへ追加する前に必ず④ 基底Spriteクラスの初期化処理を実行するようにしてください。

参照：http://westplain.sakuraweb.com/translate/pygame/Sprite.cgi#pygame.sprite.Sprite

4. main関数内の前回練習8の部分をspriteモジュールのcollide_rect関数を使用する記述に改めよ

【Advanced】練習問題

Groupクラスは，Spriteクラスのサブクラスのインスタンスを複数集約し，
まとめて処理するためのクラスである
Groupクラスを利用して，複数の爆弾を処理する記述を追加する

7. Groupクラスのコンストラクタを実行することにより，空のコンテナを作成せよ

7. 7で作成した空のコンテナに，5つの爆弾（Bombオブジェクト）を追加せよ

7. Groupクラスのインスタンスメソッドupdate()とdraw()を用いて，すべての爆弾を初期描画，移動描画せよ

7. spriteモジュールのgroupcollide関数を使用して，衝突している爆弾数を求め，1つでも衝突していたらmain関数からreturnする記述に改めよ

4限：演習課題

fight_kokaton.py

Pygameのドキュメント（日本語訳）
[http://westplain.sakuraweb.com/translate/pygame/]
やaliens.pyを参考に，「負けるな！こうかとん」を作ってみよう

※ README に，実装した機能に関する説明を記述すること
※ 新たなクラスを定義して実装すること
※ コード規約を意識して実装すること

追加機能の例：
- 効果音を導入する
- 敵キャラを導入する
- こうかとんに戦う力を授ける

ex05/README.mdも忘れずに書く

- 本日の演習で追加実装した機能についての説明を追記してみよう
  例：

```
# 第5回
## 負けるな！こうかとん（ex05/fight_kokaton.py）
### ゲーム概要
- ex05/fight_kokaton.pyを実行すると，1600x900のスクリーンに草原が描画され，
  こうかとんを移動させ飛び回る爆弾から逃げるゲーム
- こうかとんが爆弾と接触するとゲームオーバーで終了する
### 操作方法
- 矢印キーでこうかとんを上下左右に移動する
### 追加機能
- 爆弾を複数個にする
- 時間とともに爆弾が加速するor大きくなる
### ToDo（実装しようと思ったけど時間がなかった）
- [ ] 着弾するとこうかとん画像が切り替わる
### メモ
```

【再掲】別ブランチで作業しよう

3限に実装したコードを壊さないように，新たなブランチを作成し，
新ブランチで作業する

過去のadd_funcブランチが
ある場合は，
削除してから作成する
or
別のブランチ名にする

- ブランチを切る：「`git branch ブランチ名`」
- ブランチを切り替える：「`git switch ブランチ名`」

  例：「`git branch add_func`」+「`git switch add_func`」
- 別ブランチでも随時add/commitすること
- 追加機能実装が完了したら，
  コミットコメントを「4限追加機能実装完了」としてcommitすること
- そして，「`git switch main`」でmainに戻る

【再掲】ブランチをマージしよう

別ブランチでの開発がうまくいったら，mainブランチに戻り，
別ブランチでの変更履歴をmainブランチに取り込む

- ブランチをマージする：「`git merge 別ブランチ名`」
- 例：「`git switch main`」+「`git merge add_func`」

マージが完了したら，別ブランチは不要なので削除する

- ブランチを削除する：「`git branch -d 別ブランチ名`」
- 例：「`git branch -d add_func`」

そしてリモートリポジトリにプッシュしよう

5限：グループワーク

追加機能を実装後，mainブランチにmergeしたら，pushする

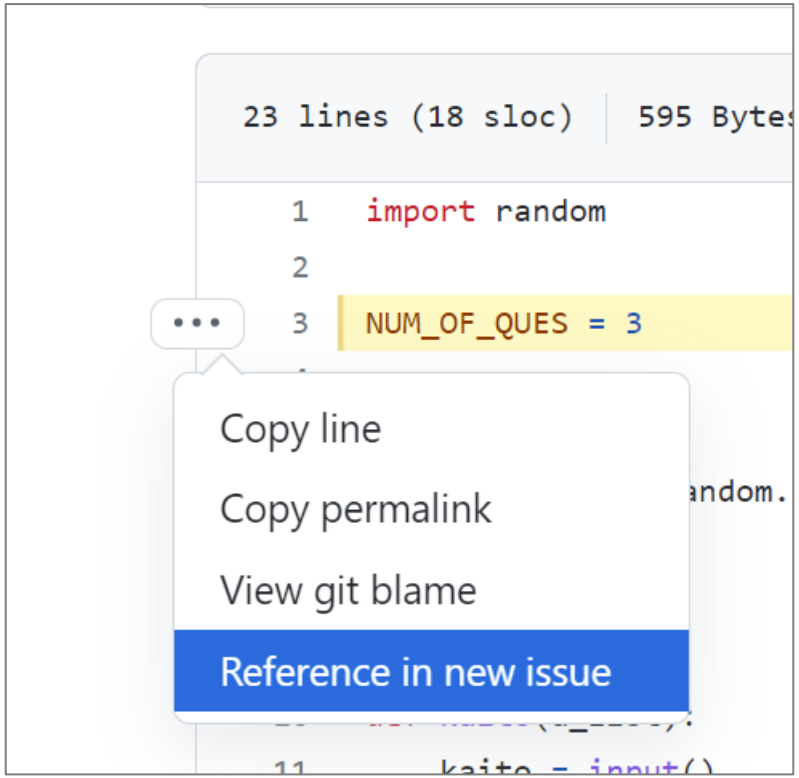
グループを作り，コードを読む

- Githubで共有されたREADME.mdとコードを読む
- ZOOMのBORでの画面共有により，追加機能をデモする
- コードについて議論する
- 説明者以外の4人のうち最低2人は，コメントをつける
  ※必ず，全員が2人以上からコメントを受けること
  ※必ず，全員が2人以上に　コメントを付けること
- Issuesでコメントを受けて，
  - 修正すべきなら修正する
  - 修正すべきでないなら，修正しない理由を返信する
- コード修正のコメントがない場合は，
  TASAからコメントを受けるように，手を上げてお願いする．

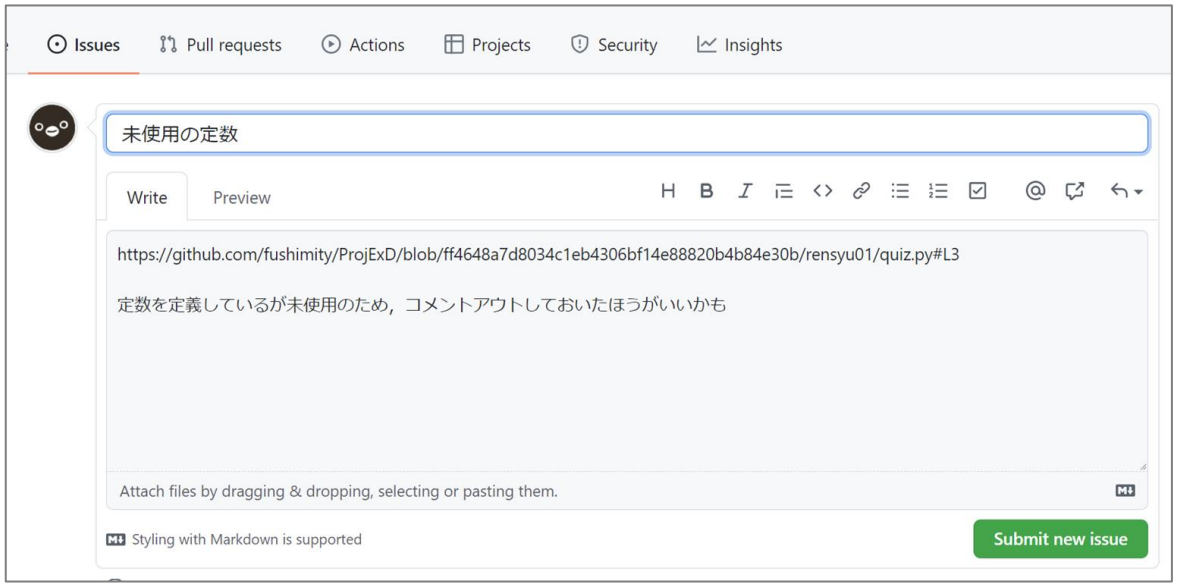
【再掲】Issueでコメントを書いてみる

- 当該コードを読み，必要に応じてコメントを送る

①コメントしたい行にカーソルを当て，「...」→「Reference in new issue」

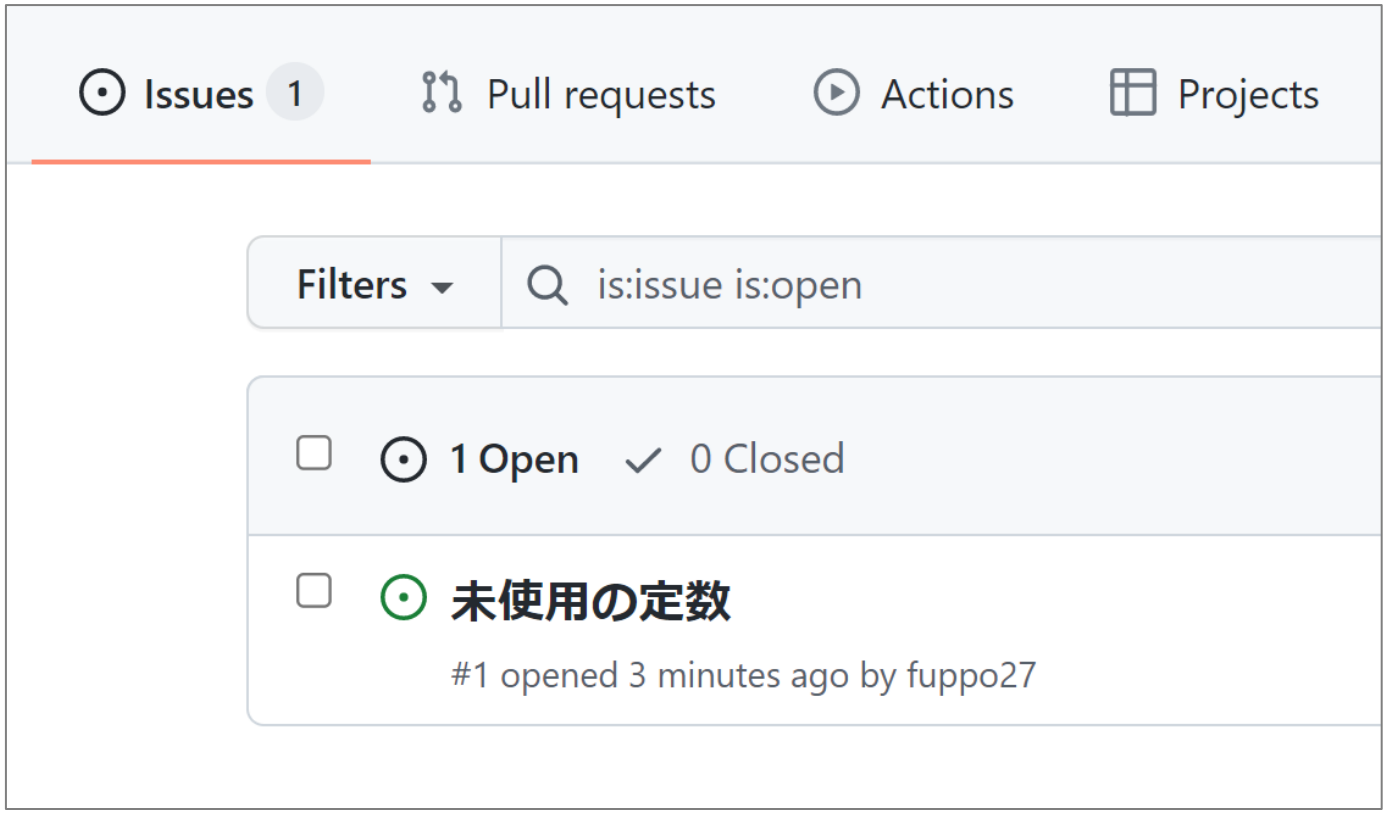


②コメントを書き，「Submit new issue」



【再掲】Issueが届く

- Issuesタブから，Issueのコメントを読み，対応する



Issueコメントを受けてコードを修正する

1. Issue#1のコメントを受けてコードを修正する

2. 修正が終わったらステージングする
   「`git add ex05/fight_kokaton.py`」

3. ステージングされた内容をコミットする
   ※重要：コミットコメントに，Issue番号を付けること

   「`git commit -m "コメント #1 に対する修正"`」

   要半角スペース

   これにより，
   Issueコメントと対応コミットがGithub上で紐づけられる
4. リモートリポジトリにプッシュする「`git push origin main`」

提出物

学籍番号は，半角・大文字で

- ファイル名：C0B21XXX_kadai05.pdf
- 内容：
  - READMEの最終版
    https://github.com/fushimity/ProjExD/blob/main/ex05/README.md
    （自分のアカウント名）
  - コードの最終版
    https://github.com/fushimity/ProjExD/blob/main/ex05/dodge_bomb.py
  - Issues一覧
    https://github.com/fushimity/ProjExD/issues

チェック項目

時間内　：100%
当日中　：　80%
今週中　：　50%
次回まで：　10%

- README の説明は十分か [0 -- 2]
  - わからない [0]，まぁわかる [1]，よくわかる [2]

- 複数の機能を追加しているか [0 -- 4]
  - TASA判断による追加機能の数に基づく．ただし4機能以上は [4]

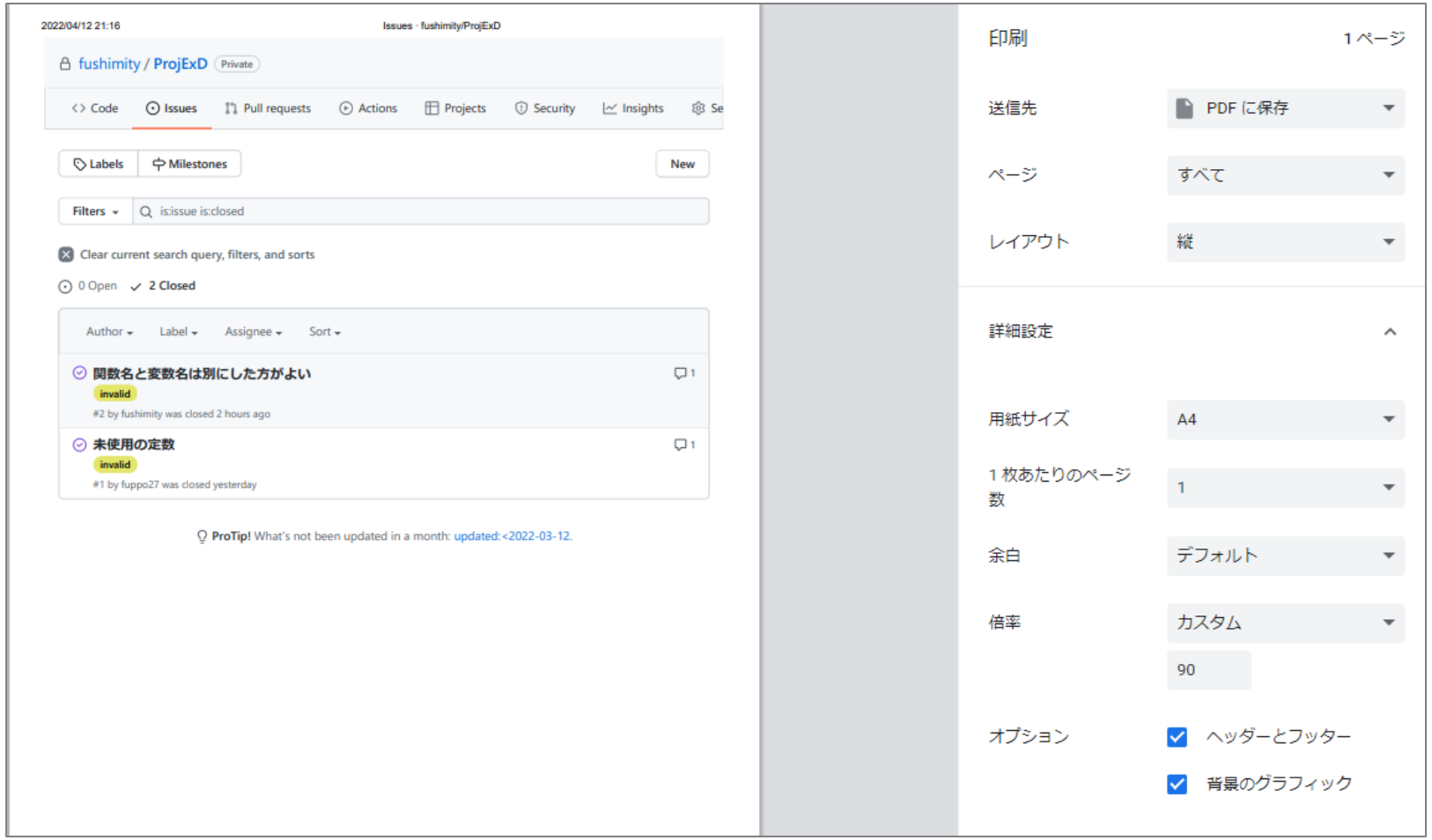
- こうかとんに戦う能力が追加されているか [0 or 1]

- クラスを新たに定義しているか [0 -- 4]
  - TASA判断による追加クラスの数に基づく．ただし4クラス以上は [4]

- fight_kokaton.pyはコード規約を考慮しているか [0 -- 4]

【再掲】ChromeでPDFとして保存する方法

1. 該当ページを表示させた状態で「Ctrl＋P」
2. 以下のように設定し，「保存」をクリックする



送信先に「Microsoft Print to PDF」は選ばない(PDFからURLの埋め込みが消える)

←送信先：PDFに保存

←ページ：すべて

←レイアウト：縦

←用紙サイズ：A4

←余白：デフォルト

←倍率：90

←両方チェック

【再掲】各PDFを単一ファイルにする方法

1. ChromeでPDFとして保存する
2. 以下のURLから各PDFをマージする
3. ファイル名を「C0B21XXX_kadai05.pdf」として保存する

- オンラインでPDFをマージ：
  https://www.ilovepdf.com/ja/merge_pdf

【再掲】提出，チェックの流れ

1. 受講生：提出物ができたらMoodleにアップロードする
2. 受講生：課題チェック依頼のスプレッドシートにて，
   待ちキューが少ないTASAの列に学籍番号を入力する
3. TASA：Moodleにアップされた課題をチェックする
4. TASA：チェックが済んだら，
   - Moodleに点数を入力する
   - スプレッドシートの学籍番号を赤字に変更する【チェック完了の合図】
5. 受講生：自分の学籍番号が赤字になったら帰る
   - チェックを待たず帰ることも可能だが，提出物に不備があっても修正できない