

プロジェクト演習 テーマD

第1回

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

授業の概要と目標

• 概要

- Pythonによるゲーム開発プログラミングを通じて、1年次に学習したPythonの基礎知識をさらに深めるとともに、実際に動くプログラムの実装を体験してもらうことを目的とする
- 読みやすい、かつ、効率的なコードを書くにはどうすれば良いかという論理的な思考を養う
- プログラムの設計と実装、成果物の発表やコードレビューをグループで実施する。これにより、技術的知識だけでなく、コミュニケーションの重要性を体験することも重視する

• 目標

- コードをわかりやすく書くことができる
- 他者のコードと自分のコードを比較し、より良いコードに修正することができる
- うまく動作した時と動作しなかった時のコードの差分を分析し、原因を突き止めることができる
- 自らが作成したプログラムの動作を説明することができる
- 他者へ意見をしたり、他者からの意見を受け入れることができる

授業の進め方

• 進め方

- 1コマ目：配布される資料を基にゲームの基本機能を実装する
- 2コマ目：各自で追加機能を実装する
- 3コマ目：グループで各人のコードをレビューする
- 各回授業終了時に、その日に実装したレビュー前のプログラム、レビュー後のプログラム、レビュー内容をまとめたレポートを提出する

• 教科書・参考書

- 教科書：授業時（1週間前に公開するように頑張ります）に配布する講義資料
- 参考書：

- 廣瀬 豪「Pythonでつくる ゲーム開発 入門講座」ソーテック社
- 廣瀬 豪「Pythonでつくる ゲーム開発 入門講座 実践編」ソーテック社



前提知識と評価方法

- 前提知識（準備学習）

- 1年次に学習したPythonの基礎（制御構文，関数，クラス，例外処理など）を復習しておくこと。
- テーマDの各回の内容は，それまでに学習した内容を理解していることを前提としている。
したがって，次回の講義までに前週の内容を復習することが必須である。
- 不明な点の質問は担当教員のオフィスアワーで対応する。
- Moodleには講義資料を事前に公開するので，資料内容をもとに予習してから講義に臨むこと。

- 評価の方法

テーマC（50%）とテーマD（50%）を合計して成績を決定する。

< テーマD >

- 第1回～第5回の提出物（ $15 \times 5 = 75\%$ ），第6回の共同開発成果物（15%），第7回の成果発表（10%）
- 提出物のフィードバックはMoodleを経由して提供する。

授業の流れ

- 第1回:実験環境の構築 / Pygameの基礎 / Gitの基礎
- 第2回:Pygameによるゲーム開発の基礎 / コード規約とコードレビュー
- 第3回:オブジェクト指向によるゲーム開発 / GitHubの応用
- 第4回:Pygameによるゲーム開発の応用 / 共同開発の基礎
- 第5回:共同開発演習（個別実装）
- 第6回:共同開発演習（共同実装）
- 第7回:共同開発演習（成果発表）

本日のお品書き

1. 実験環境の構築

1. 仮想環境：Anaconda
2. 言語：Python
3. ライブラリ：Pygame
4. エディタ：VScode
5. バージョン管理：Git for windows, GitHubアカウント

2. Pygameの基礎

3. Gitの基礎

4. Pygameの演習

目標：Pygameの概要を理解し、
簡単なコードを実装でき、GitHubにプッシュできる

グループ分け

- 5人1組のグループを作成する
- 次回～最終回まで同じメンバーで活動する
- 活動内容
 - 第2, 3回：メンバーが実装したコードを共有し, コメント, 修正をし合う
 - 第4, 5, 6回：メンバーと共同開発する
 - 第7回：メンバーと発表準備をし, 登壇し発表する
- Python力が高い人, 低い人が混ざったグループが望ましい
- グループができたなら, 以下のスプレッドシートに学籍番号を入力する
(Moodle講義ページの上部にもリンクあり)

※4人, 6人になってしまった場合は, 要相談.
基本的に16グループになるように調整します.

<https://docs.google.com/spreadsheets/d/1F8o547bqU0oHb0L4ViR6FrDk0nr09piVtkdv8MMowjo/edit#gid=0>

3限：実験環境の構築

想定する実験環境

- OS : Windows 11 (HomeでもEducationでもOK)
- 仮想環境 : Anaconda (Navigator $\geq 2.3.2$)
- 言語 : Python $\geq 3.10.9$ (Pygame $\geq 2.1.2$)
- エディタ : Visual Studio Code $\geq 1.75.1$
- ローカルリポジトリ : Git for Windows $\geq 2.40.0$
- リモートリポジトリ : GitHub

その他の実験環境でも構いませんが、何か問題があった場合の対処は自己責任でお願いします。

仮想環境の構築

- 他の授業の環境を壊さないように、プロ演D用の仮想環境を用意する
- Anaconda promptで、「ProjExD」という名前の仮想環境を構築する

```
C:¥Users¥fsmtkys> conda create -n ProjExD
```

- 構築できたか確認する

```
C:¥Users¥fsmtkys> conda info -e

# conda environments:
#
base                * C:¥Users¥fsmtkys¥anaconda3
ProjExD             C:¥Users¥fsmtkys¥anaconda3¥envs¥ProjExD
```

- アクティベートする

```
C:¥Users¥fsmtkys> conda activate ProjExD
```

Python 3.10.9のインストール

- Anaconda promptで以下を入力しインストールする

```
(ProjExD) C:¥Users¥fsmtkys>conda install python=3.10.9
```

- インストールされたものを確認する

```
(ProjExD) C:¥Users¥fsmtkys>conda list
# packages in environment at C:¥Users¥fsmtkys¥anaconda3¥envs¥ProjExD:
#
# Name                                Version                                Build      Channel
bzip2                                1.0.8                                he774522_0
ca-certificates                      2023.01.10                            haa95532_0
certifi                              2022.12.7                            py310haa95532_0
libffi                                3.4.2                                hd77b12b_6
openssl                              1.1.1s                               h2bbff1b_0
pip                                  22.3.1                              py310haa95532_0
python                              3.10.9                               h966fe2a_0
:
zlib                                  1.2.13                               h8cc25b3_0
```

Pythonライブラリのインストール

- Pygame 2.1.2をインストールする

```
(ProjExD) C:¥Users¥fsmtkys>pip install pygame==2.1.2
```

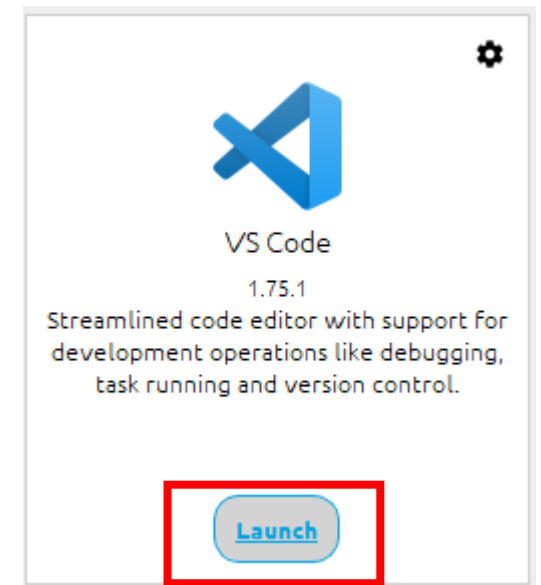
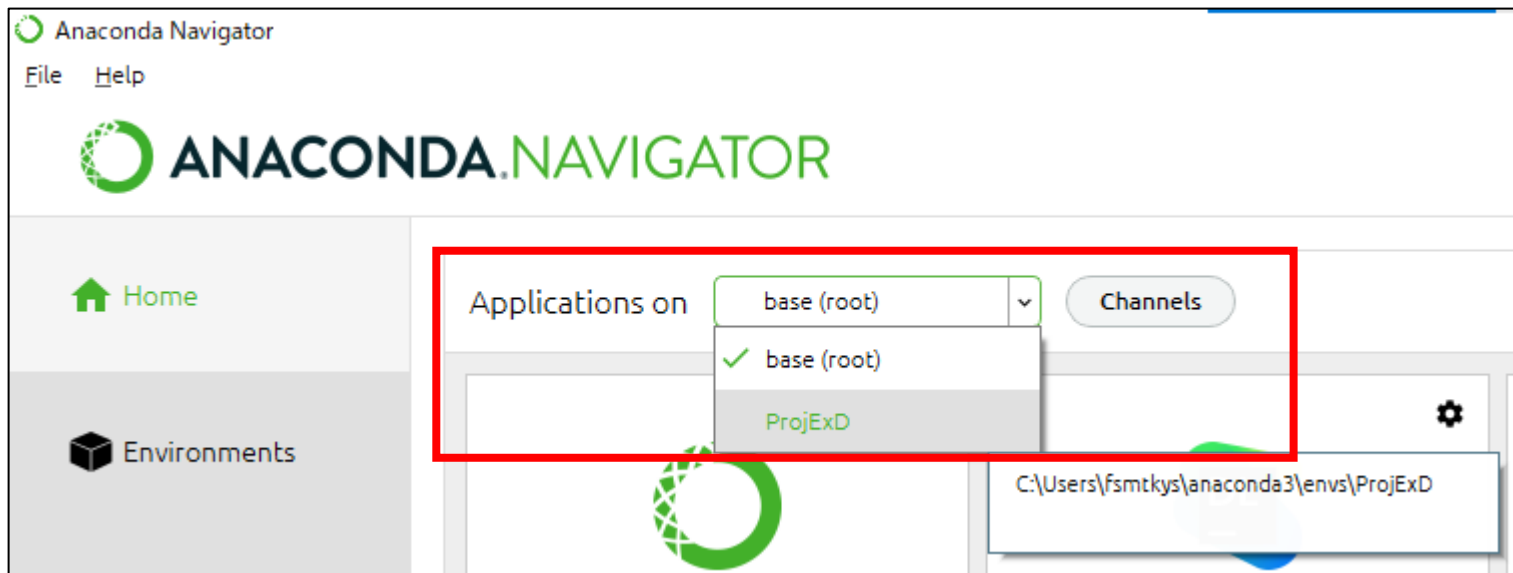
※注意：condaではなくpipでインストール

- 確認する

```
(ProjExD) C:¥Users¥fsmtkys>conda list
# packages in environment at C:¥Users¥fsmtkys¥anaconda3¥envs¥ProjExD:
#
# Name          Version        Build    Channel
pygame          2.1.2          pypi_0   pypi
pygments        2.14.0         pypi_0   pypi
```

VScodeを開く

- Anaconda navigatorで、仮想環境「ProjExD」を選択し、VS CodeをLaunchする



拡張機能のインストール

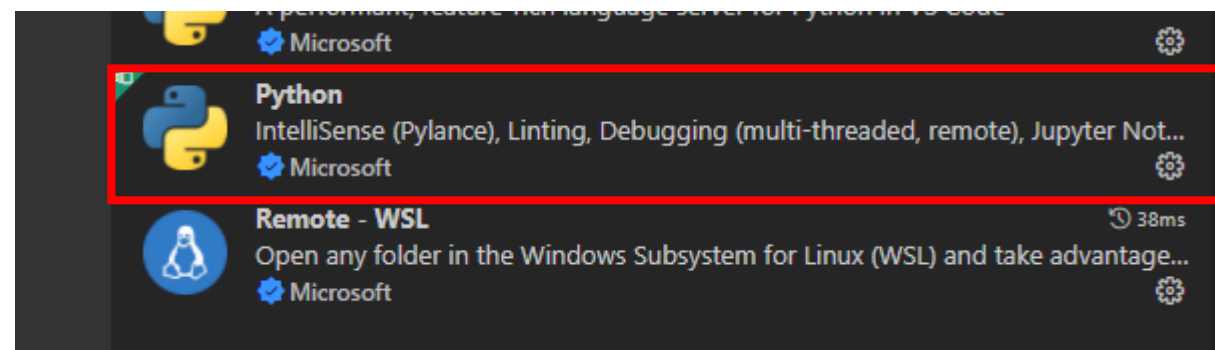
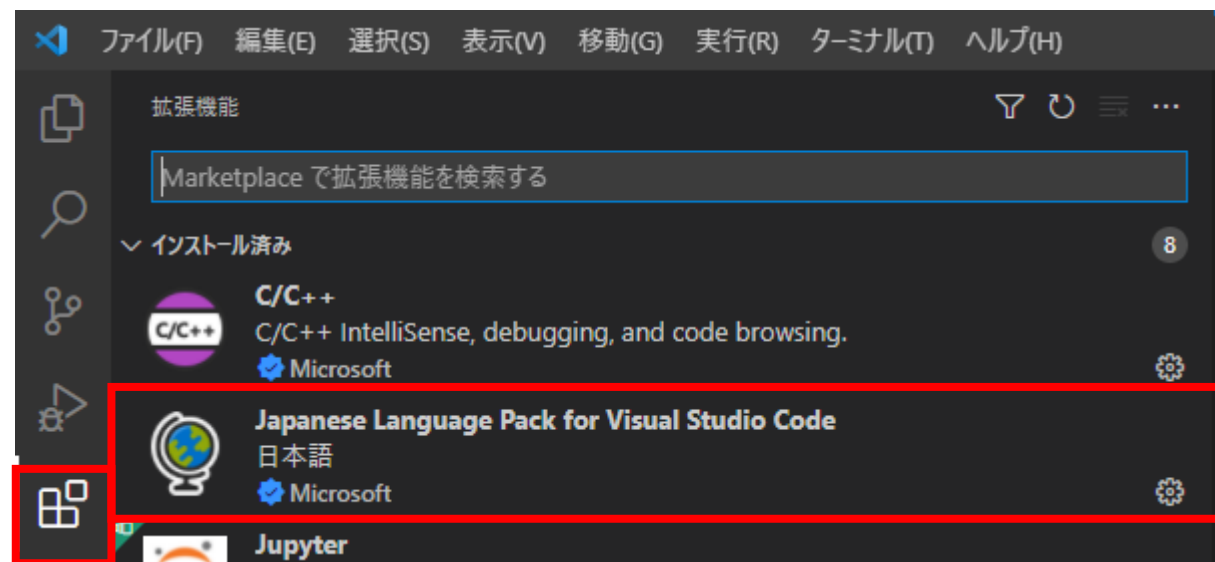
- Japanese Language Pack
 - Python
- をインストールする

ついでに, Vimが苦手な人は

- Vim

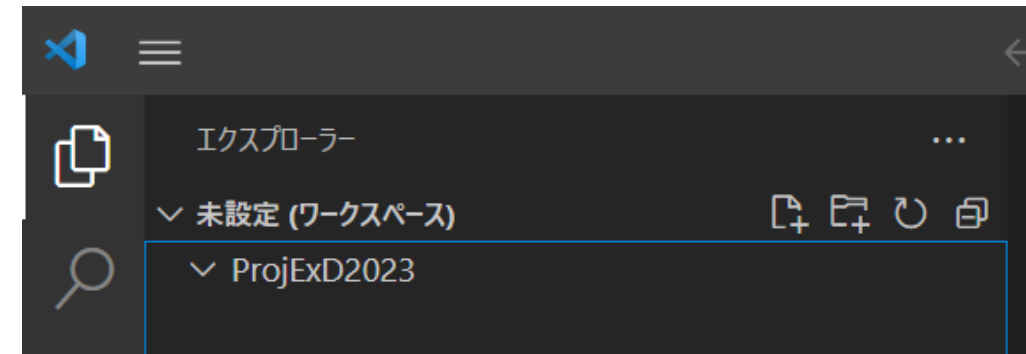
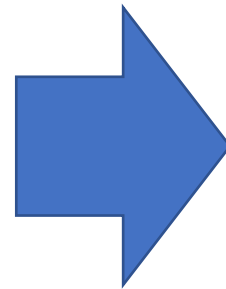
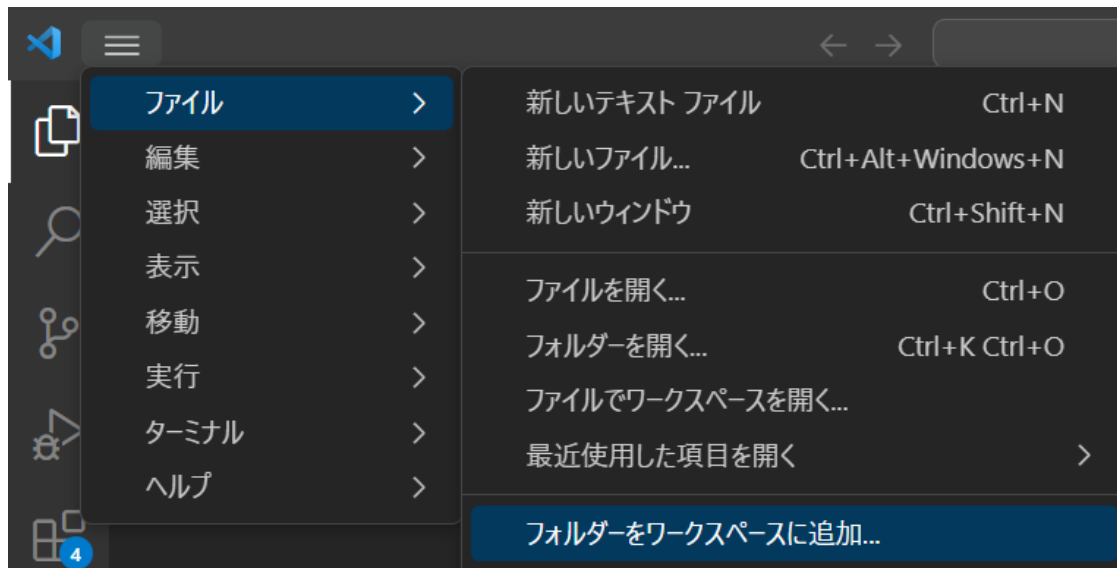
のキーマップを無効にする

※日本語環境にならない人は
VScodeを再起動してみよう



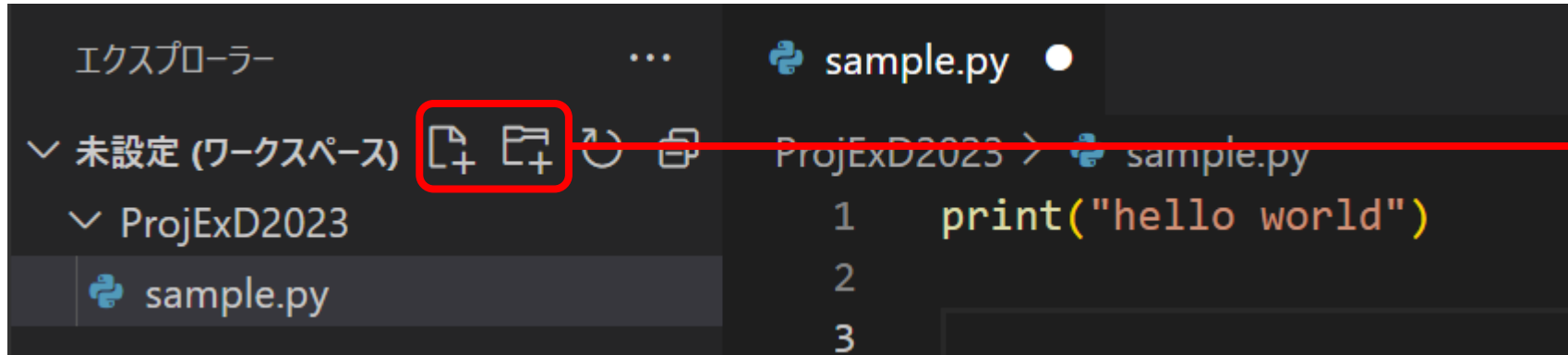
作業フォルダのワークスペースへの追加

- 任意の場所（DocumentsやDesktop）に「ProjExD2023」を作成する
- VScodeの「ファイル」→「フォルダーをワークスペースに追加」から、上記フォルダを追加する





サンプルプログラムの作成

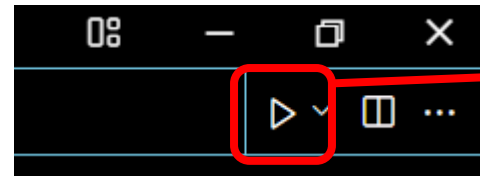
- 「ProjExD2023」フォルダの下に「sample.py」を作成する




ファイル作成
フォルダ作成のアイコン

- 「Ctrl+S」で保存して, 「Ctrl+F5」で実行する

- 保存前  sample.py ●
- 保存後  sample.py ×

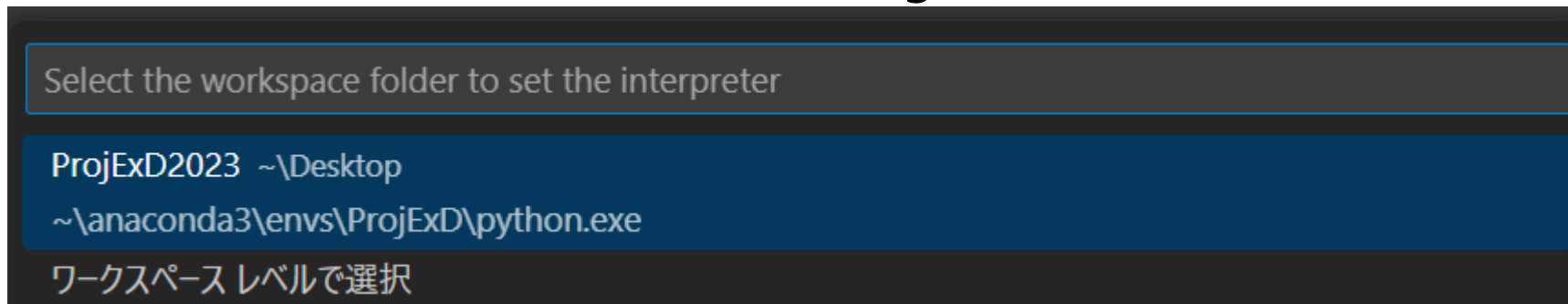


右上の▶アイコンでも実行できる

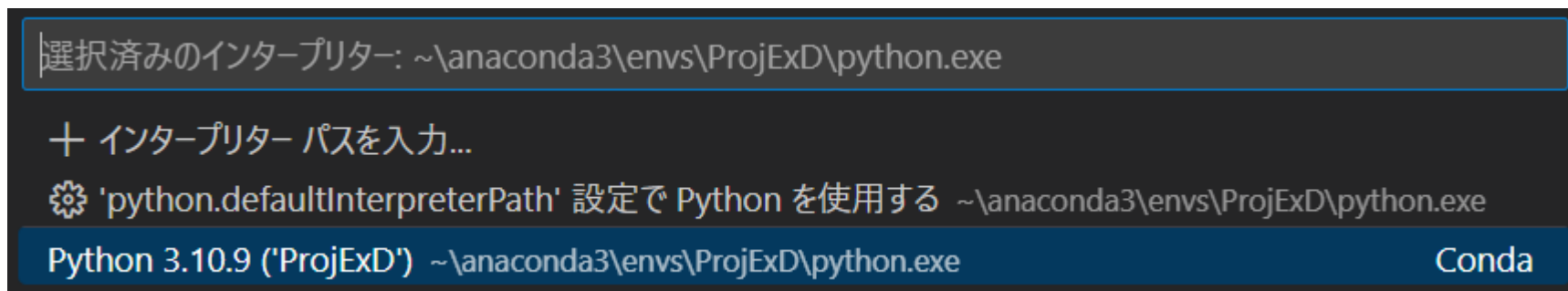
- 実行後は右下が  Python 3.10.9 ('ProjExD': conda) になっているはず
- ※うまくいかなかったら, VScodeを一度閉じてみよう

実行できなかったら → VScode右下を確認

- Python 3.10.9 ('ProjExD': conda) となっておらず,
「Select...」またはPythonのバージョンだけになっている場合は,
クリックして,
- ワークスペースフォルダ「ProjExD2023」を選ぶ

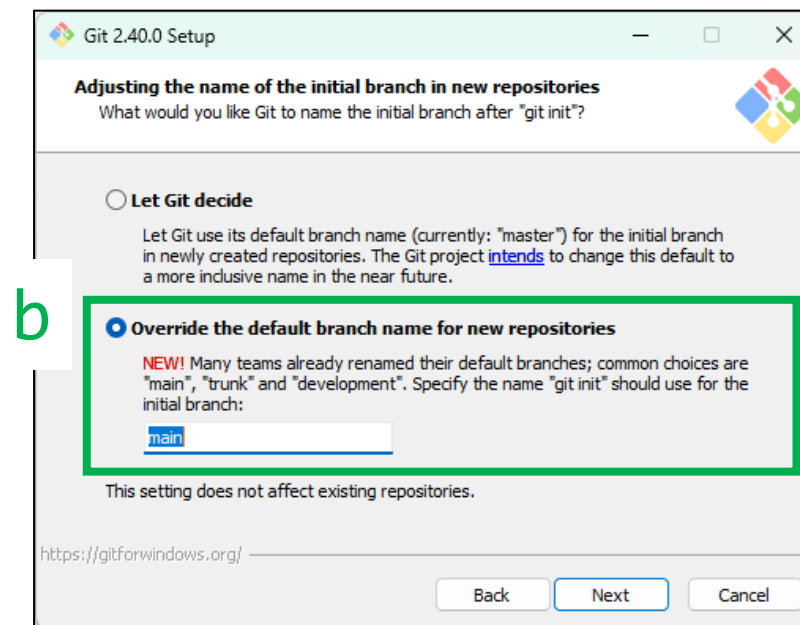
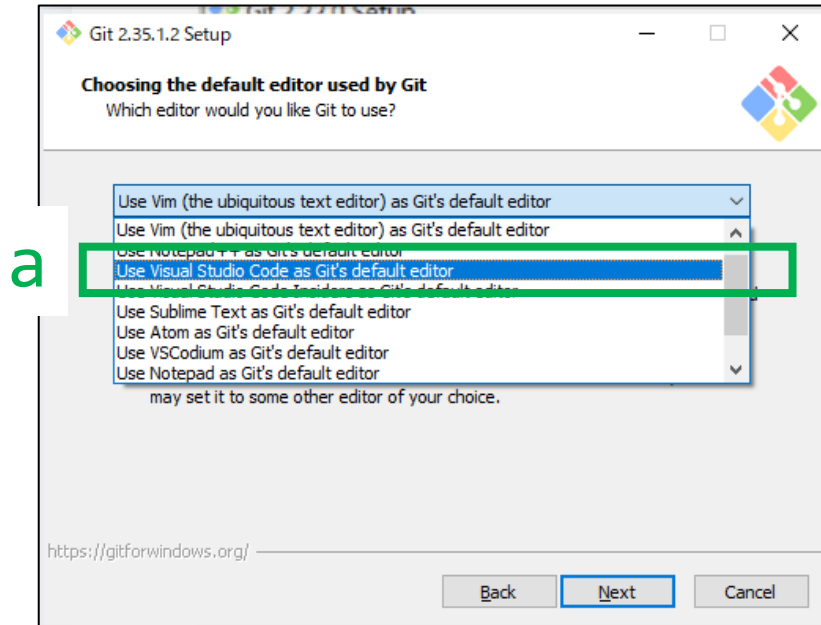


- インタプリタパス「Python3.10.9 ('ProjExD')」を選ぶ



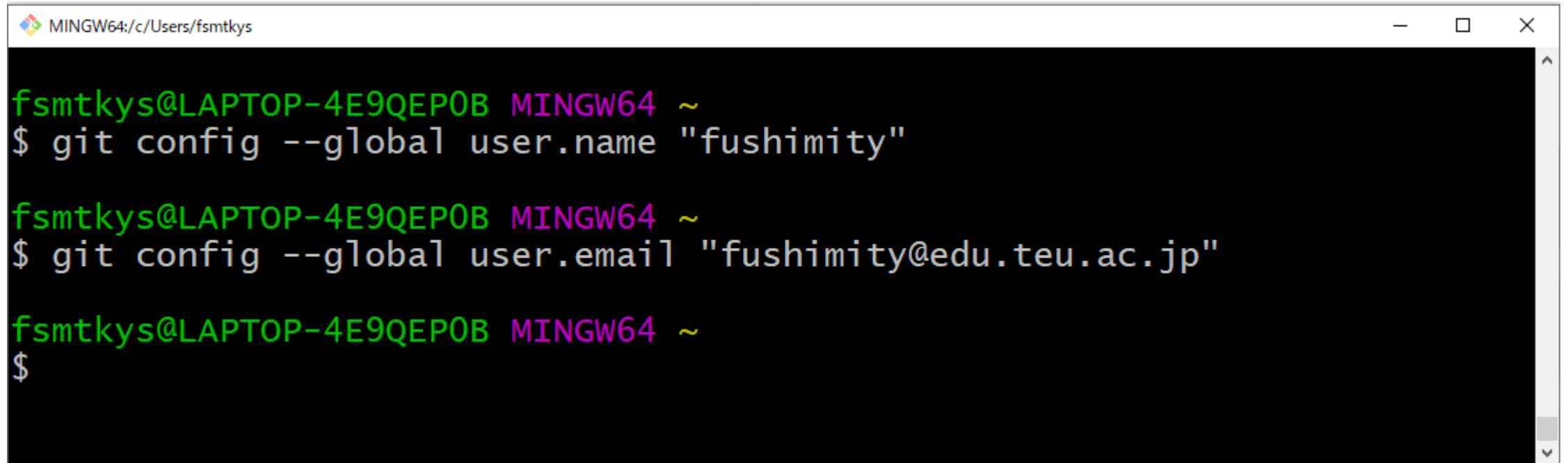
Git for Windowsのインストール

- <https://gitforwindows.org/>からインストーラーをDLする
- 以下の2点を除くすべてで「NEXT」をクリックする
 - a. **エディタ選択画面**：「Use Visual Studio Code as Git's default editor」を選択する
 - b. **デフォルトブランチ指定画面**：「Override the default branch name for new repositories」で「main」とする



Gitの初期設定

- Git Bashにてユーザ名とメールアドレスを設定する
 - user.name : わかりやすい名前 (今後表示されるので, 変なのにしらないこと)
 - user.email : 大学のメールアドレス



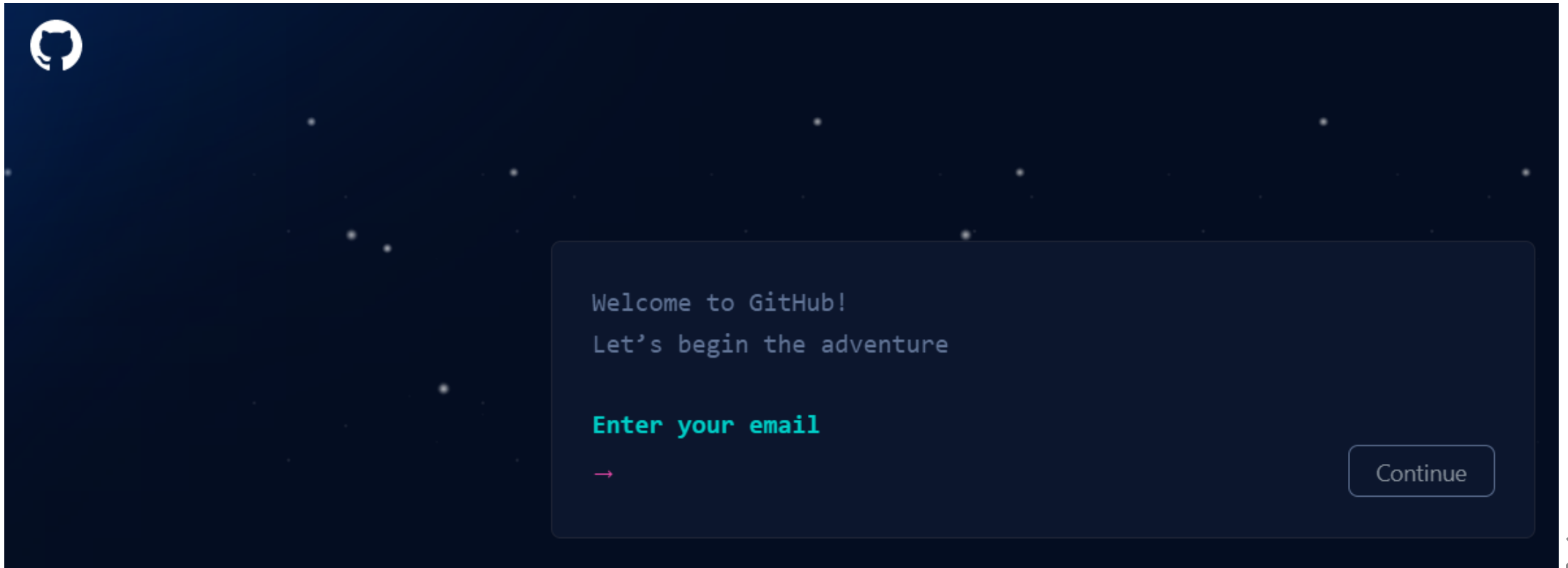
```
MINGW64:/c/Users/fsmtkys
fsmtkys@LAPTOP-4E9QEP0B MINGW64 ~
$ git config --global user.name "fushimity"

fsmtkys@LAPTOP-4E9QEP0B MINGW64 ~
$ git config --global user.email "fushimity@edu.teu.ac.jp"

fsmtkys@LAPTOP-4E9QEP0B MINGW64 ~
$
```

GitHubアカウント

- <https://github.com/>にて, 「Sign up」する
- Enter your emailに大学のメールアドレスを入力する
- 無料版のプランを選択し, アカウントを作成する



4限： Pygameの基礎

PyGameドキュメント（日本語訳）：

<http://westplain.sakuraweb.com/translate/pygame/>

配布物の確認

- Moodleからex01.zipをDLし, ProjExD2023フォルダの下に配置

【配布物配置後のディレクトリ構造】

- ProjExD2023/

- sample.py

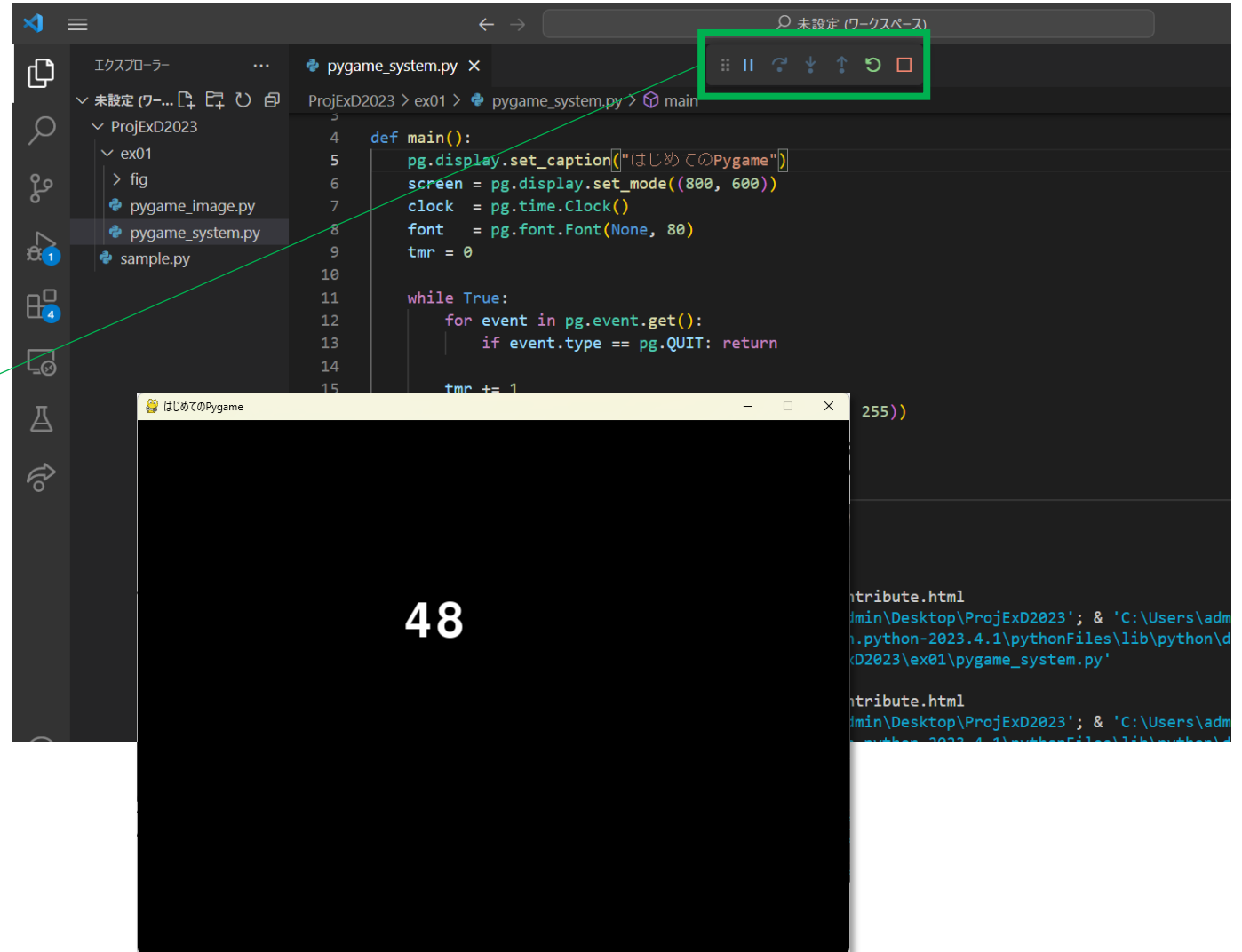
- ex01/


本日の配布物

- pygame_system.py . . . pygameのテンプレ
 - pygame_image.py . . . はばたけ！こうかとん
 - japanese_font.py . . . 日本語フォント表示方法
 - fig/
 - pg_bg.jpg . . . 背景画像
 - {0, ..., 9}.png . . . こうかとん画像

まずは、pygame_system.pyを動かしてみる

- 「Ctrl+S」で保存する
- 「Ctrl+F5」で実行する



- 「」で終了する

初期化と終了

- pygameパッケージ（モジュール）をimportする

```
import pygame as pg
```

- pygameモジュールを初期化する

```
pg.init()
```

- pygameモジュールの初期化を解除する

```
pg.quit()
```

その後、プログラムを「sys.exit()」で終了する

```
if __name__ == "__main__":  
    pg.init()  
    main()  
    pg.quit()  
    sys.exit()
```

「if __name__ == "__main__"」は
他ファイルからimportされた時は発動せず
コマンドラインから実行された時に発動する。

Pygameを用いたコードの全体像

1. Pygameモジュールの初期化：`pg.init()`

2. ゲームの初期化（ループに入る前）

main関数

- 画面の表示
- プレイヤー, 敵などの配置
- スコアの初期化

3. ゲームのループ（無限ループ）

- キャラクターの移動
- 戦闘処理
- スコアの計算
- 画面の更新

4. ゲーム終了処理（ループを抜ける）

- スコア画面の表示

5. Pygameモジュール終了：`pg.quit()`, `sys.exit()`

使用しているモジュール

- displayモジュール：画面制御や画面Surfaceを生成する
- fontモジュール：文字描画
- drawモジュール：図形描画
- timeモジュール：ゲームにおける時刻を制御する
- eventモジュール：キーボードイベントなどを取得する

【重要な概念（クラス）】

- Surfaceクラス
- Rectクラス

displayモジュール

- set_caption関数：画面のタイトルを設定する
 - 引数：タイトル文字列
- set_mode関数：画面用のSurfaceインスタンスを生成する
 - 引数：幅と高さのタプル
 - 戻り値：画面Surface
- update関数：画面を更新する
- 使用例

```
pg.display.set_caption("はじめてのPygame")
screen = pg.display.set_mode((800, 600))

pg.display.update()
```

Surfaceクラス

- Pygameでは、読み込んだ画像、作成した図形、文字などはSurfaceクラスのインスタンスとして表現され、別のSurfaceに転送する（＝貼り付ける）ことで表示する
- 大元のスクリーン用のSurfaceに、プレイヤー画像などのSurfaceを貼り付けることで、ゲーム画面を描画する
- **blitメソッド**：自身に別のSurfaceを貼り付ける
 - 第1引数：別のSurfaceインスタンス
 - 第2引数：位置座標を表すリストやタプル, **Rectインスタンス**
- **get_rectメソッド**：Surfaceが存在する範囲を取得する
 - 戻り値：範囲を表す**Rectインスタンス**
- **fillメソッド**：Surfaceを一色に塗りつぶす
 - 引数：色タプル

fontモジュール

- Fontクラス：文字列のフォントを規定するクラス
 - コンストラクタ：フォントオブジェクトを生成する
 - 引数：フォント名, サイズ
 - renderメソッド：指定色の文字列を書いたSurfaceインスタンスを生成する
 - 引数：文字列, True, 色タプル
 - 戻り値：文字Surface
- 使用例

```
fonto = pg.font.Font(None, 80)
txt = fonto.render("hello",
                   True, (255, 255, 255))

screen.blit(txt, [300, 200])
```

- ・ ・ ・ フォントサイズを80に設定する
- ・ ・ ・ 白字で"こんにちは"と書かれたSurfaceインスタンスを生成する
- ・ ・ ・ 文字Surfaceを画面Surfaceに転送する = 貼り付ける

- 日本語を表示する際には, 日本語フォントを指定する (japanese_font.py参照)

drawモジュール

- circle関数：円を生成する
 - 引数：描画用Surface, 色, 中心座標, 半径
- line関数：直線の線分を生成する
 - 引数：描画用Surface, 色, 始点座標, 終点座標
- rect関数：四角形を生成する
 - 引数：描画用Surface, 色, 四角形の範囲
- polygon関数：多角形を生成する
 - 引数：描画用Surface, 色, 点リスト

• 使用例

```
enn = pg.Surface((20, 20))
pg.draw.circle(enn, (255, 0, 0), (10, 10), 10)
enn.set_colorkey((0, 0, 0))
```

- • • 一辺が20の正方形Surfaceの
- • • 中心に半径10の赤い円を描画
- • • 黒を透過させる

Surfaceを画面に表示する方法

- 手順 0 : スクリーンSurfaceの作成する

```
screen = pg.display.set_mode((800, 600))
```

- 手順 1 : { 画像, 図形, フォント } Surfaceの作成する

```
img = pg.image.load("ex01/fig/3.png")
```

・ ・ ・ 画像Surface

```
enn = pg.Surface((20, 20))
```

・ ・ ・ 図形用の空Surface

```
pg.draw.circle(enn, (255, 0, 0), (10, 10), 10)
```

・ ・ ・ 空Surfaceに図形を描く

```
txt = fonto.render("hello", True, (255, 255, 255))
```

・ ・ ・ 文字Surface

- 手順 2 : SurfaceをスクリーンSurfaceに転送 (blit) する

```
screen.blit(img, [300, 200])
```

- 手順 3 : 画面を更新する

```
pg.display.update()
```

Surfaceを転送する（blit）方法

- 座標リスト，タプルを使用する場合

移動させない場合

```
screen.blit(img, [300, 200])
```

- Rectを使用する場合

移動させる場合

```
img_rct = img.get_rect()  
img_rct.center = 300, 200  
screen.blit(img, img_rct)
```

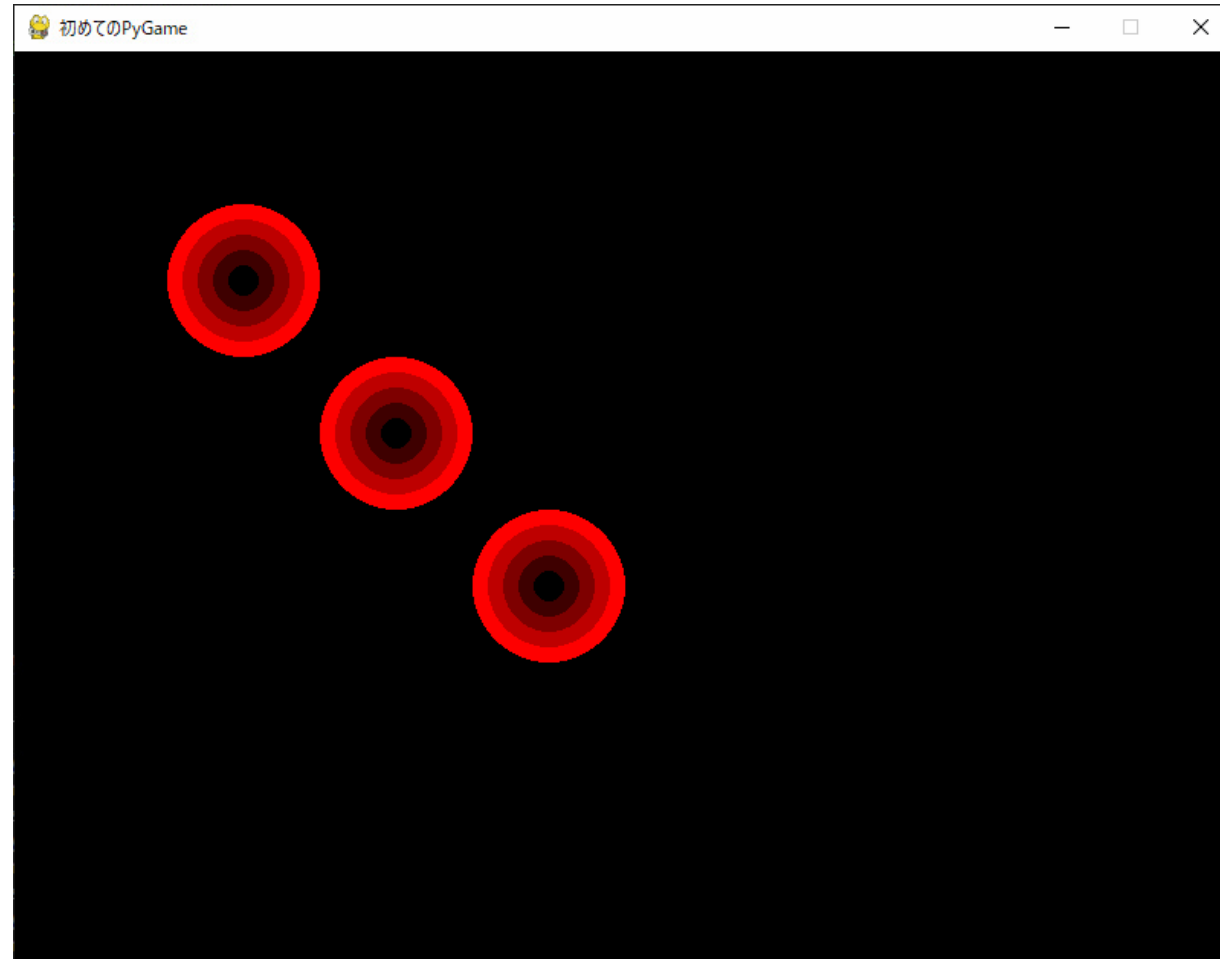
- ・ ・ ・ 画像Surfaceに対応する画像Rectを取得する
- ・ ・ ・ 中心座標を300, 200に設定する例
- ・ ・ ・ 画像SurfaceをスクリーンSurfaceにRectに従って貼り付ける

Rectクラス

- Pygameでは、Rectクラスのインスタンスを使用して、Surface（画面、画像、図形、文字など）を描画する矩形範囲を設定、変更する
- インスタンス変数
 - 位置に関するもの：top, left, bottom, right, center, centerx, centery
 - 大きさに関するもの：size, width, height
- インスタンスメソッド
 - move_ip：移動させる
 - 引数：横方向速度, 縦方向速度
 - colliderect：別のRectオブジェクトと重なっているか判定する
 - 引数：別のrectオブジェクト
 - 戻り値：重なっていればTrue／重なっていなければFalse

【おまけ】 blitによるSurface合成の例

`draw.circle`関数により生成した色とサイズの異なる複数の円を重ねて表示



【おまけ】 blitによるSurface合成の例

```
draw_sfc = pg.Surface((100,100))
```

←描画用Surface (幅 : 100, 高さ : 100) を生成する

```
pg.draw.circle(draw_sfc, (255,0,0), (50,50), 50)
pg.draw.circle(draw_sfc, (191,0,0), (50,50), 40)
pg.draw.circle(draw_sfc, (127,0,0), (50,50), 30)
pg.draw.circle(draw_sfc, ( 63,0,0), (50,50), 20)
pg.draw.circle(draw_sfc, (  0,0,0), (50,50), 10)
```

←描画用Surfaceに
赤(255,0,0)で
横 : 50, 縦 : 50に
半径50の円を描画する

←描画用Surfaceに
黒(0,0,0)で
横 : 50, 縦 : 50に
半径10の円を描画する

```
scrn_sfc.blit(draw_sfc, (100,100))
scrn_sfc.blit(draw_sfc, (200,200))
scrn_sfc.blit(draw_sfc, (300,300))
```

←描画用Surfaceであるdraw_sfcを
スクリーンSurfaceであるscrn_sfcの
横 : 300, 縦 : 300に描画する

timeモジュール

- Clockクラス：時間管理，計測用のクラス
 - コンストラクタ：時間計測用オブジェクトを生成する
 - tickメソッド：指定したフレーム秒の遅延を発生させる

※遅延させないと，高速に処理されてしまう（∵CPUは高速である）

- 引数：フレームレート

フレームレート（frame per second）：
動画において，単位時間（1秒）あたりに
処理させるフレームすなわち「コマ」の数を示す頻度の数値のこと

• 使用例

```
clock = pg.time.Clock()  
clock.tick(1)
```

．．． 1秒あたり1フレーム処理するように遅延させる

```
clock.tick(10)  
clock.tick(0.1)
```

．．． 1秒に10フレーム（＝0.1秒に1フレーム）

．．． 1秒に0.1フレーム（＝10秒に1フレーム）

eventモジュール

- get関数：イベントキューから全てのイベント情報を取得する

- 戻り値：Eventインスタンスが並んだリスト

- イベント（キー入力，マウスクリックなど）は，発生した順番にイベントキューに追加される
 - get関数により取得されたEventインスタンスは，イベントキューから削除される

- type属性：イベントの種類を表すインスタンス変数

- QUIT：×ボタンのクリック

- KEYDOWN：キーの押下

- key属性：キーの種類を表すインスタンス変数

- 使用例

```
for event in pg.event.get():  
    if event.type == pg.QUIT:  
        return
```

- . . . イベントキュー取り出した1つのイベントに対して
 - . . . そのイベントが「×」ボタンクリックだったら
 - . . . returnする

その他のモジュール

- imageモジュール：画像描画
- transformモジュール：画像変換
- keyモジュール：キーボードイベント
- mouseモジュール：マウスイベント

imageモジュール / transformモジュール

- image.load関数：指定したパスのファイルを読み込み，画像Surfaceを生成する
 - 引数：ファイルパス
 - 戻り値：画像Surface
- transform.flip関数：上下左右を反転
 - 引数：画像Surface, 左右反転bool, 上下反転bool
 - 戻り値：変換後の画像Surface
- transform.scale関数：拡大縮小
 - 引数：画像Surface, 幅高さタプル
 - 戻り値：変換後の画像Surface
- transform.rotate関数：回転
 - 引数：画像Surface, 回転角度
 - 戻り値：変換後の画像Surface
- transform.rotozoom関数：回転 + 拡大縮小
 - 引数：画像Surface, 回転角度, 倍率
 - 戻り値：変換後の画像Surface
- 使用例

← rotateよりrotozoomの方が変換後の画像がきれい

```
kk_img = pg.image.load("ex01/fig/3.png")
kk_img = pg.transform.flip(kk_img, True, False)
kk_img = pg.transform.rotozoom(kk_img, 10, 1.0)
```

- • • 3.pngを読み込み
- • • 左右反転させ
- • • 10度反時計回りに回転

keyモジュール / mouseモジュール

- `key.get_pressed`関数：すべてのキーの押下状態を取得する
 - 戻り値：押下状態のリスト
 - リストのインデックス：キーボード定数（`K_UP`, `K_SPACE`, `K_a`, `K_0`, `K_LSHIFT`, `K_F1` など）
 - リストの値：True or False

• 使用例

```
key_lst = pg.key.get_pressed()
if key_lst[pg.K_UP]:
    kk_rct.move_ip((0, -1))
```

- ・ ・ ・ キーの押下状態リストを取得し
- ・ ・ ・ 上矢印が押されていたら
- ・ ・ ・ こうかといふRectを横に0, 縦に-1移動させる

- `mouse.get_pos`関数：マウスカーソルの位置を返す
 - 戻り値：ウィンドウ枠内での位置座標タプル（横座標, 縦座標）

4限：Gitの基礎

ローカルリポジトリの初期化

- 起動しているGit bashを開く
- ディレクトリを移動する

```
admin@LAPTOP-63464884 MINGW64 ~  
$ pwd  
/c/Users/admin  
  
admin@LAPTOP-63464884 MINGW64 ~  
$ cd Desktop  
  
admin@LAPTOP-63464884 MINGW64 ~/Desktop  
$ cd ProjExD2023/ex01/
```

- gitリポジトリを初期化する：`git init`

```
admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01  
$ git init  
Initialized empty Git repository in C:/Users/admin/Desktop/ProjExD2023/ex01/.git/  
  
admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01 (main)
```

↑
ブランチ名が `(main)` になっていることを確認

ステージング, コミットしてみよう

- ステージングする : `git add ファイル名`
- コミットする : `git commit -m "コメント"`

```
admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01 (main)
$ git add pygame_system.py

admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01 (main)
$ git commit -m "初コミット"
[main (root-commit) 15506bc] 初コミット
1 file changed, 27 insertions(+)
create mode 100644 pygame_system.py
```

- 確認する : `git log --oneline`

```
admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01 (main)
$ git log --oneline
15506bc (HEAD -> main) 初コミット
```

ステージング : ファイルをインデックスに追加すること
インデックスにはファイルの変更内容が記録される
ステージングされたファイルのみがコミットの対象となる

コミット : Gitリポジトリに変更内容を登録すること

ちなみに①

- 作業ディレクトリとステージングエリアの状態を確認する：

`git status`

```
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    fig/
    pygame_image.py

nothing added to commit but untracked files present (use "git add" to track)
```

- Untracked files：同じフォルダ内にあるけど，addしてないからgitが追跡できていないファイルたち
- nothing added to commit：変更が登録されてないので何もコミットしないよ

ちなみに②

- 追跡 (track) 中のpygame_system.pyに変更を加えてみると

```
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   pygame_system.py

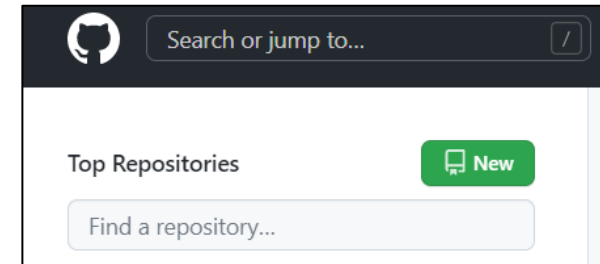
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        fig/
        pygame_image.py

no changes added to commit (use "git add" and/or "git commit -a")
```

- Changes not staged for commit : 変更あったけどステージングされてないよ (コミットするならaddでステージングしてね)

リモートリポジトリの作成


- ブラウザでGitHubのトップページを開き, **New**をクリックする
 - Repository nameを「ProjExD_01」とする
 - Descriptionを「プロジェクト演習D（1回目）」とする
 - Visibilityを「Public」にする
- Create repository**をクリックする



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)


Owner * / Repository name *


 fushimity / ProjExD_01 ✓

Great repository names are short and memorable. Need inspiration? How about [super-duper-spork?](#)

Description (optional)

プロジェクト演習D（1回目）

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

できあがり



Search or jump to...



Pull requests

Issues

Codespaces

Marketplace

Explore

fushimity / ProjExD_01 Public

Pin

Unwatch 1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTPS

SSH

`https://github.com/fushimity/ProjExD_01.git`

← コピーする

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# ProjExD_01" >> README.md
```

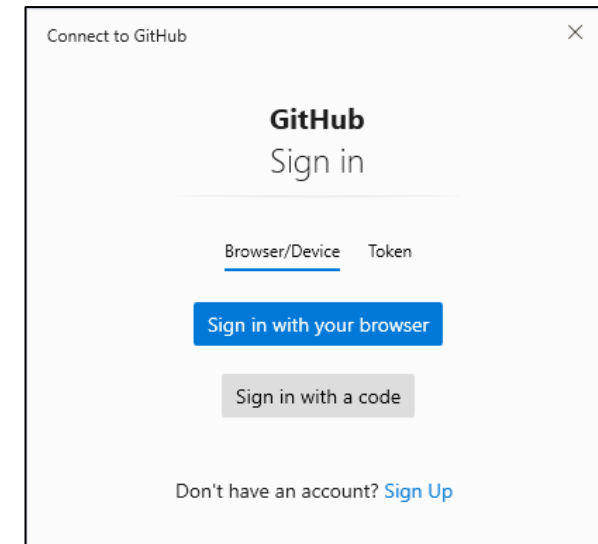
リモートリポジトリの登録とプッシュ

※Git bashでは
Shift+Insで
貼り付け可能

- リモートリポジトリを登録する：`git remote add origin gitURL`
- リモートリポジトリ(origin)へプッシュする：`git push origin main`

```
MINGW64/c/Users/admin/Desktop/ProjExD2023/ex01
admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01 (main)
$ git remote add origin https://github.com/fushimity/ProjExD_01.git

admin@LAPTOP-63464884 MINGW64 ~/Desktop/ProjExD2023/ex01 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 572 bytes | 572.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/fushimity/ProjExD_01.git
 * [new branch]      main -> main
```



- 途中、認証画面が現れたら **Sign in with your browser** をクリックする

プッシュ：ローカルリポジトリのコミット履歴をリモートリポジトリに送信すること

リモートリポジトリの修正 ※間違えた場合のみ

- リモートリポジトリの登録内容を確認する：`git remote -v`
- リモート名(origin)を間違えた場合
`git remote rename 今のリモート名 新しいリモート名`
- URLを間違えた場合
`git remote set-url origin 正しいURL`

エラーが出てプッシュできない場合

- 権限エラーやnot foundエラーの場合
 - `git remote set-url origin https://gitアカウント名@github.com/gitアカウント名/リポジトリ名.git`
- 「Support for password authentication was removed on August 13, 2021....」エラーの場合
 - Git for Windowsのバージョンが古い可能性がある
 - アップデートをする：`git update-git-for-windows`
 - プッシュし直したら、認証用のウィンドウが出るはず

リモートリポジトリの確認

- ページを更新すると

The screenshot shows the GitHub interface for a repository named 'fushimity / ProjExD_01'. The repository is public. The 'Code' tab is selected, showing the 'main' branch with 1 branch and 0 tags. A commit by 'fushimity' is displayed, titled '初コミット' (First commit), with the commit hash '15506bc' and a timestamp of '8 minutes ago'. The commit message is 'pygame_system.py 初コミット'. Below the commit, there is a prompt to 'Add a README' to help people understand the project.

GitHub interface showing the repository **fushimity / ProjExD_01** (Public).

Navigation tabs: **Code** (selected), Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings.

Repository details: **main** (1 branch, 0 tags). Buttons: **Go to file**, **Add file**, **Code**.

Commit history:

- fushimity** 初コミット (15506bc, 8 minutes ago, 1 commit)
- pygame_system.py (初コミット, 8 minutes ago)

Footer: Help people interested in this repository understand your project by adding a README. **Add a README**

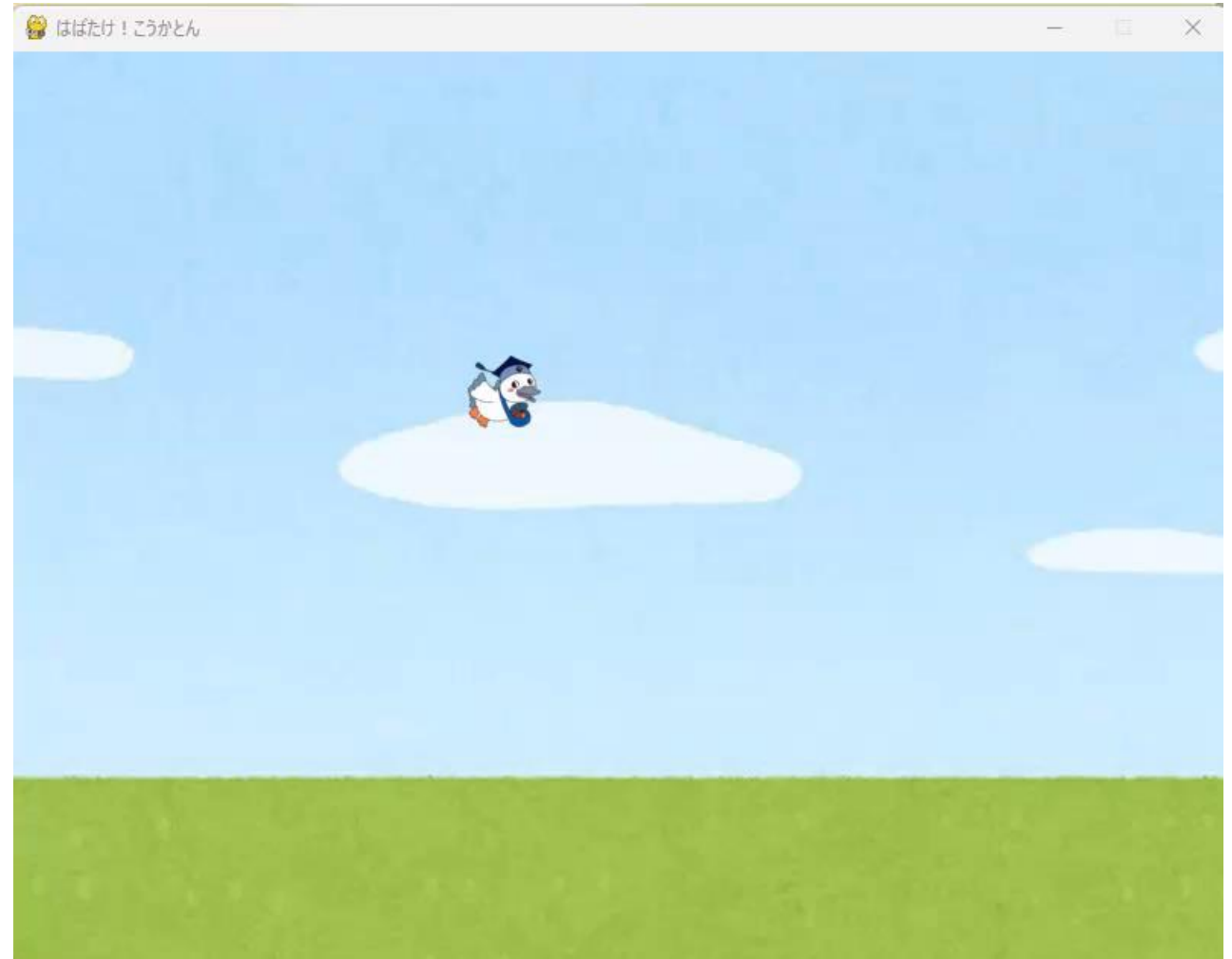
※これが表示されれば、プッシュは成功している

Pygameの基礎（練習）

pygame_image.py

「はばたけ！こうかとん」を実装しよう

- 野原の上空に**こうかとん**が颯爽とはばたく~~ゲーム~~
- 実装する機能
 - 背景画像の描画
 - こうかとん画像の描画
 - こうかとんがはばたくエフェクト



練習問題

※1問ずつステージング, コミットしよう

【ゲームの初期化】

1. 幅1600×高さ900の背景画像「pg_bg.jpg」を読み込み, Surfaceを生成せよ.
2. こうかどん画像「3.png」を読み込み, Surfaceを生成せよ.
そして, 左右を反転せよ.
3. 2.の画像Surfaceと, 10度回転させた画像Surfaceを要素とするリストを生成せよ.

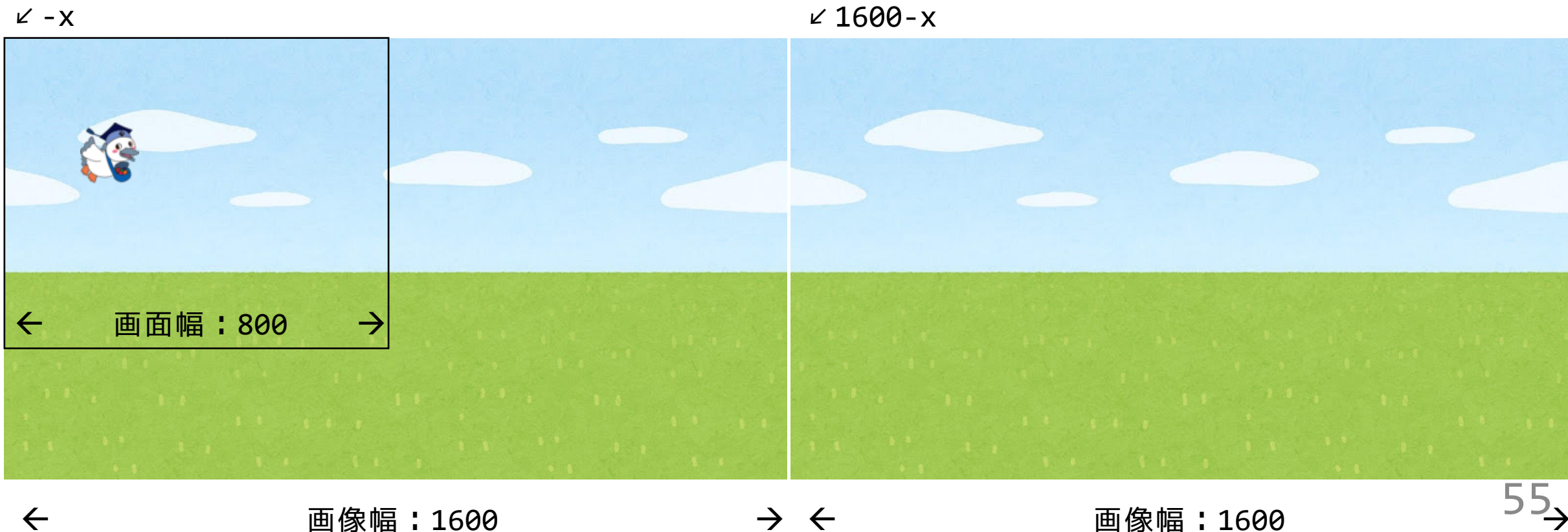
【ゲームのループ】

4. 背景画像を表示せよ.
5. 横300, 縦200の位置に3.のリストの画像Surfaceを貼り付けよ.
ただし, タイムステップに応じて交互に表示させるようにせよ.

練習問題（つづき）

遅いと思ったら、「`clock.tick(10)`」
を「`clock.tick(100)`」に変更しよう

6. こうかとんが画面右に向かって進んでいるように見せるために、
背景画像を右から左に動くように、背景画像の横座標を修正せよ。
- 2つの背景画像Surfaceを以下のようにblitする
 - x を、 $0 \rightarrow 1 \rightarrow \dots \rightarrow 1599 \rightarrow 0 \rightarrow 1 \rightarrow \dots \rightarrow 1599$ のように繰り返す値とする



5限：Pygameの演習

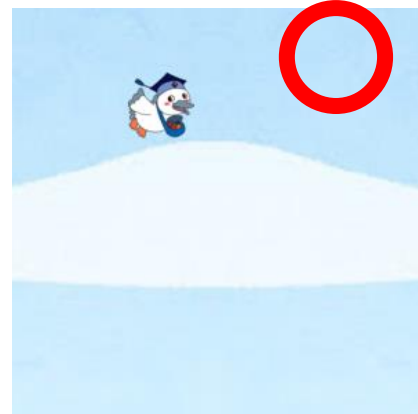
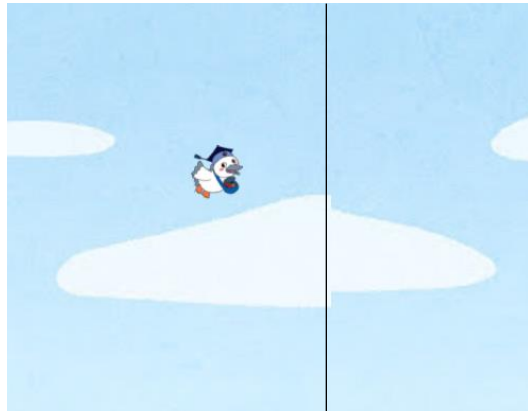
演習課題：「はばたけ！こうかとん」の改良

1. こうかとんが滑らかに羽ばたくように修正せよ。

- 練習問題の時より滑らかならOK

2. 背景画像の雲が不連続になっている部分を修正せよ。

※新しいpg_bg.jpgを作るのではなく、python, pygameの機能を利用すること



3. 修正後のコードをプッシュせよ。

4. 次ページ以降にしたがって、提出物を作成，提出せよ。

提出物

学籍番号は, 半角・大文字で

- ファイル名 : C0A22XXX_kadai01.pdf
- 内容 : 以下の順番でPDFを結合して提出すること
 - コミット履歴
https://github.com/fushimity/ProjExD_01/commits/main
 - コードの最終版
https://github.com/fushimity/ProjExD_01/blob/main/pygame_image.py

自分のアカウント名

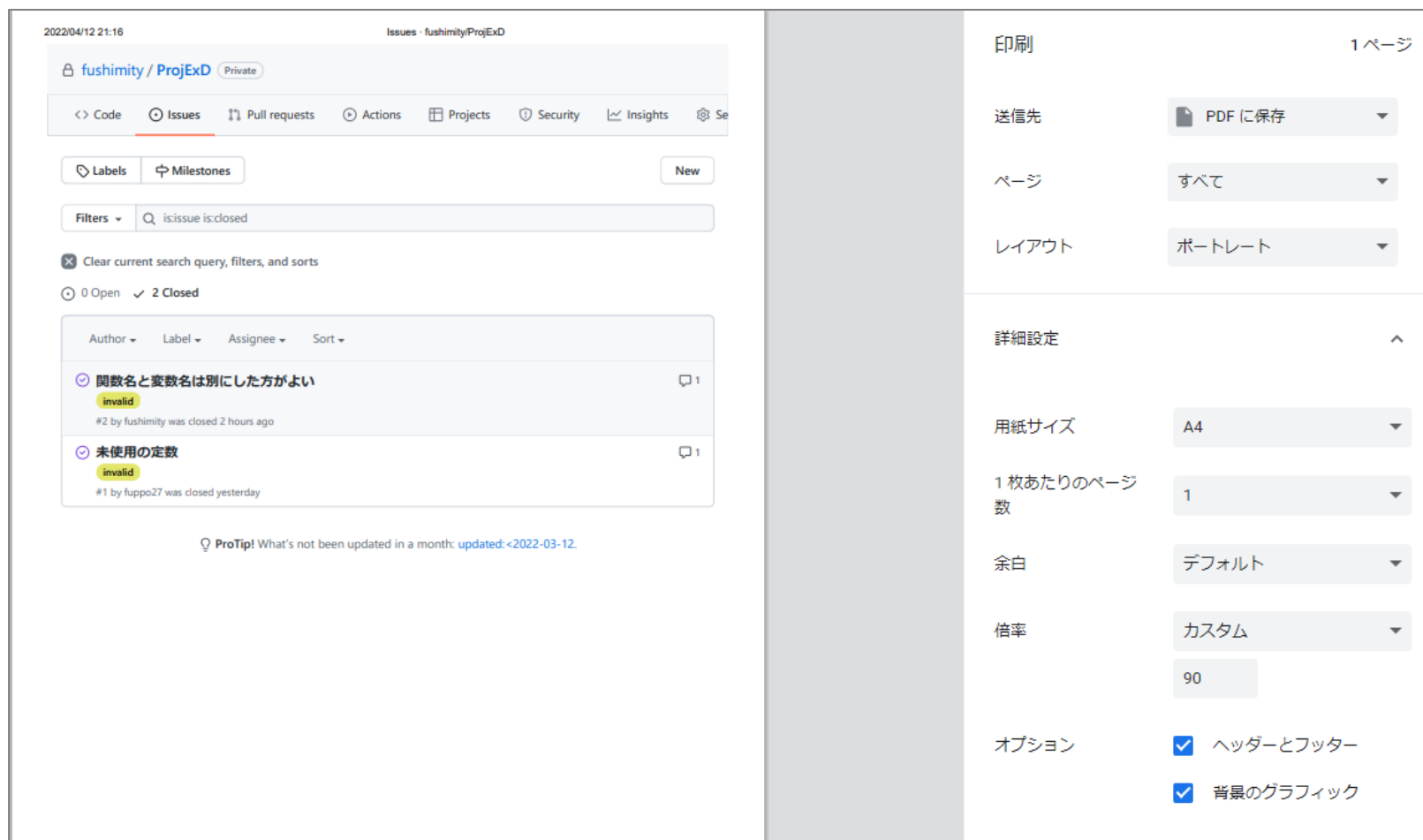
提出物の作り方

※スクショ画像は認めません。
以下の手順に従ってPDFを作成し、提出すること

1. ChromeでPDFとして保存する（次ページを参照）
2. 以下のURLから各PDFをマージする
https://www.ilovepdf.com/ja/merge_pdf
3. ファイル名を「C0A22XXX_kadai01.pdf」として保存する

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：PDFに保存

←ページ：すべて

←レイアウト：ポートレート

←用紙サイズ：A4

←余白：デフォルト

←倍率：90

←両方チェック

チェック項目

1. 提出物のルールが守られている [0 ~ 4]
 - a. ファイル名
 - b. スクショでない
 - c. クリッカブル
 - d. PDFの順番 (コミット履歴→pygame_image.pyの最終版)
2. 複数回コミットし, コミット履歴がプッシュされている [0 ~ 2]
 - 練習問題のみ: 1点
 - 演習課題も: 2点
3. こうかとんが滑らかに羽ばたいている [0 or 3]
4. 背景の雲が不連続でない [0 or 3 or 6]
 - 手つかず: 0点
 - ワープしたり, ぼやけたりする: 3点
 - 完璧: 6点

チェックの手順

※基本的に再提出できません。どうしてもの場合は要相談。

1. 受講生：提出物（pdf）を作成し，Moodleに提出する
 2. 受講生：担当TASAに成果物（ゲーム）を見せに行く
 3. TASA：提出物とゲームのデモを確認し，点数を確定する
 4. 受講生：帰る
 5. FSM：近日中に課題と点数を確認し，Moodleに登録する
- 時間内にチェックが終わらなそうな場合は，提出物をMoodleに提出し帰る
（次回までor次回の3限にチェックされる）

← 時間外提出扱いになり
割引いて採点するので
できるだけチェックを
受けること