6. ミニプロジェクト

https://www.kkaneko.jp/cc/enshu2/index.html

金子邦彦

BY NC SA

① 実践的問題解決能力

やりたいこと自ら探求し、Pythonを用いて解決策を模索。問題解決スキルを身につける。

② プログラミング技術の深化:

ミニプロジェクトを通じて、Pythonの具体的な経験を積み、重要なプログラミング技術を実践的に学ぶ。

③ 自主性と達成感の促進:

自分のアイディアを Python プログラムとして実現し、問題を解決する。その過程で感じる創造の喜びと達成感が、自信を育み、学習への意欲を高める。

アドバイス

実践的問題解決能力の向上

- 自分で「実現してみたいこと」を探し、Pythonで解決策を探る
- 実世界の課題に即したスキルの習得を目指す

プログラミング技術の深化

- 経験を重ねる
- 間違いの修正、改善を粘り強く繰り返す

自主性と達成感の促進

- アイディアを Python のプログラムとして実現
- 創造と成功を通じて自信と学習意欲を向上

仲間との会話も極めて重要です

6-1. イントロダクション

プログラミング

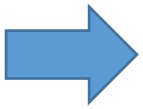
- **プログラム**を設計し作成するプロセス（プログラミング）は，**創造的な活動**
- **アイデアを形にできる**ことが，**プログラミング**の魅力

プログラミングの目的

- **プログラム**は，**コンピュータ**に指示を出し，所定の作業を遂行させる
- 複雑な作業も**自動化**し，効率化することが可能

```
a = [200, 400, 300]
for i in a:
    print (i * 1.08)
```

Python プログラムの
ソースコード



```
216.0
432.0
324.0
```

プログラムの
実行結果

Python

- **Python** は多くの人々に利用されている**プログラミング言語**の１つ
- **読みやすさ，書きやすさ，幅広い応用範囲**が特徴

```
from keras.models import Sequentia
model = Sequential()
from keras.layers import Dense, Ac

model.add(Dense(units=64, input_di
model.add(Activation('relu'))
model.add(Dense(units=max(set(y_tr
model.add(Activation('softmax'))
model.compile(loss='sparse_categor
              optimizer='sgd',
              metrics=['accuracy']
model.fit(x_train, y_train, epochs
score=model.evaluate(x_test, y_tes
print(score)
model.predict(x_test)
model.summary()
```

```
och 1/200
'3 [==============================] - 0s
 3200
 och 2/200
 3 [==============================] - 0s
 200
 ch 3/200
  [==============================] - 0s
 0
```

コマンドによる Python プログラム実行

① Python プログラムの保存

```
x = 100
if (x > 20):
    print("big")
else:
    print("small")
s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)
```

作成した **Python プログラム**の**ソースコード**を，例えば「foo.py」という名前の**ファイルに保存**

② Python プログラムの実行

```
kaneko@www:/tmp$ python foo.py
big
15
```

プログラムを実行するには，シェル（例えば，Windows の場合はコマンドプロンプト）を開き，「python foo.py」のようなコマンドで実行

trinket

- **Trinket** は**オンライン**の **Python**、**HTML** 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

- **Python の標準機能**を登載、その他、次のモジュールやパッケージがインストール済み

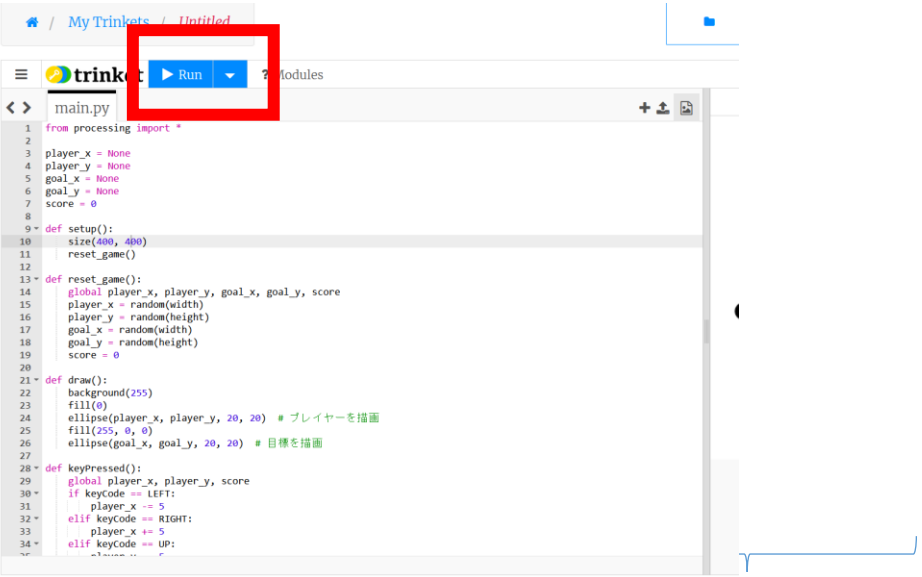
math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

trinket でのプログラム実行

- **trinket** は **Python, HTML などのプログラムを書き実行できる**サイト

https://trinket.io/python/d61d36633c

のように、違うプログラムには違う URL が割り当てられる



実行、STOP ボタン

ソースコードの
メイン画面

実行結果

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習
trinket での
Python プログラムの実行

【トピックス】

- **trinket**
- **Python プログラムのソースコード**
- **Python プログラムの実行**
- **AI のプログラム作成能力**

① trinket の次のページを開く

https://trinket.io/python/d61d36633c

② 実行結果が，次のように表示されることを確認

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

/ My Trinkets / enshu2-6

trinket ▶ Run ? Modules

main.py

```python
from processing import *

player_x = None
player_y = None
goal_x = None
goal_y = None
score = 0

def setup():
    size(400, 400)
    reset_game()

def reset_game():
    global player_x, player_y, goal_x, goal_y, score
    player_x = random(width)
    player_y = random(height)
    goal_x = random(width)
    goal_y = random(height)
    score = 0

def draw():
    background(255)
    fill(0)
    ellipse(player_x, player_y, 20, 20)  # プレイヤーを描画
    fill(255, 0, 0)
    ellipse(goal_x, goal_y, 20, 20)  # 目標を描画

def keyPressed():
    global player_x, player_y, score
    if keyCode == LEFT:
        player_x -= 5
    elif keyCode == RIGHT:
        player_x += 5
    elif keyCode == UP:
```

Result

プログラムの目的

- **キーボードの矢印キー**を使用して、**画面上の目的地にオブジェクトを移動**。

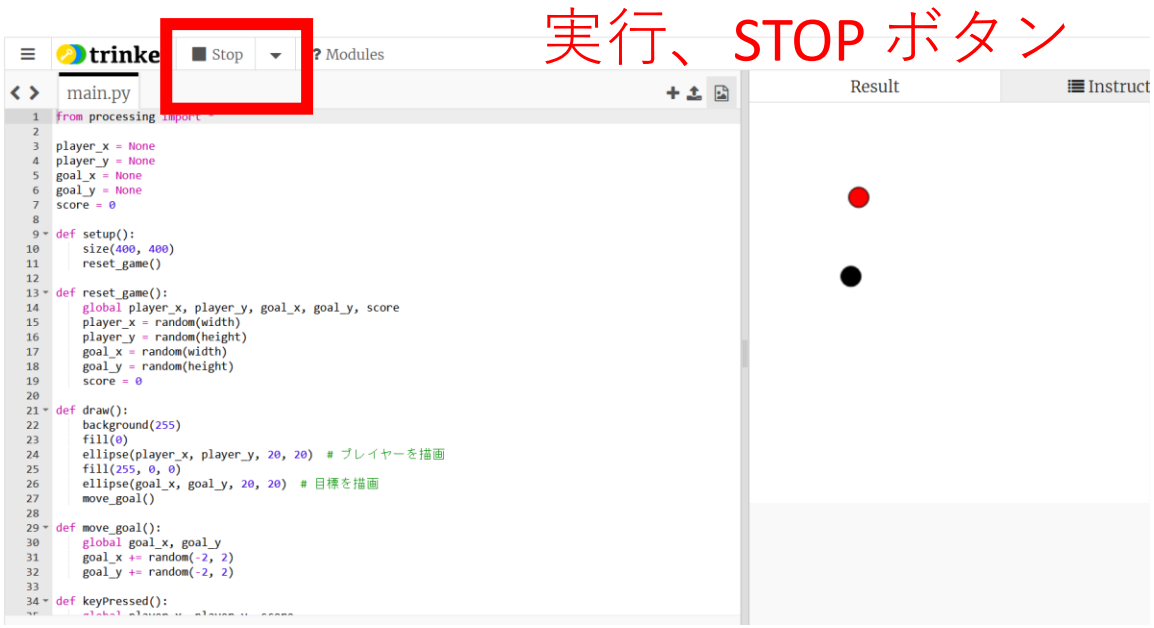
学習の目的

- Trinket の基本操作（プログラムの実行開始と停止、編集、プログラム共有）と能力の理解
- このプログラムは、**AI (ChatGPT 3.5)** に**「Python の processing を用いて面白いゲームを作成してください」**のプロンプトを与えて**生成されたプログラムコードをそのまま使用**している。
- AIによるプログラム作成の能力を理解する。

③ trinket の次のページを開く

https://trinket.io/python/ab4ac351b3

④ 実行結果が，次のように表示されることを確認



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

/ My Trinkets / *Untitled*

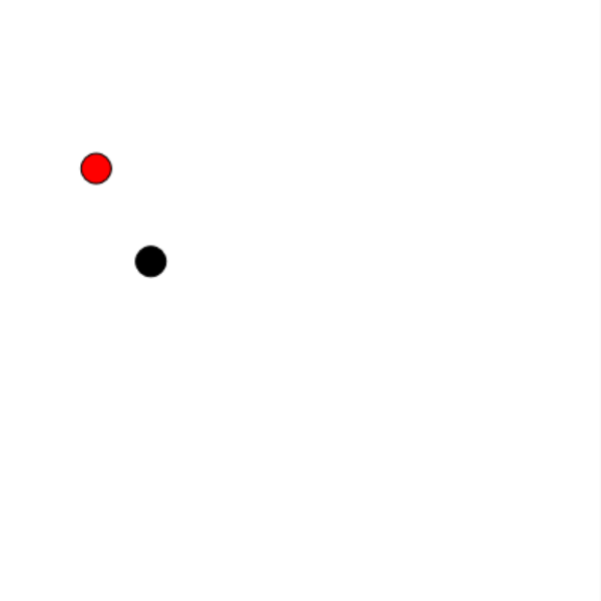
Copy | Share | Save

trinket | Stop | ? Modules

main.py

```python
from processing import *

player_x = None
player_y = None
goal_x = None
goal_y = None
score = 0

def setup():
    size(400, 400)
    reset_game()

def reset_game():
    global player_x, player_y, goal_x, goal_y, score
    player_x = random(width)
    player_y = random(height)
    goal_x = random(width)
    goal_y = random(height)
    score = 0

def draw():
    background(255)
    fill(0)
    ellipse(player_x, player_y, 20, 20)  # プレイヤーを描画
    fill(255, 0, 0)
    ellipse(goal_x, goal_y, 20, 20)  # 目標を描画
    move_goal()

def move_goal():
    global goal_x, goal_y
    goal_x += random(-2, 2)
    goal_y += random(-2, 2)

def keyPressed():
```

Result | Instructions



- これは、**AI (ChatGPT 3.5) に「次のプログラムについて、画面上の目標が自動で動くようにしてください」**のプロンプトと、前のプログラムのソースコードを与えたものである
- AIによる**プログラム改善**の能力を知る

次のプログラムについて、画面上の目標が自動で動くようにしてください。from processing import * player_x = None player_y = None goal_x = None goal_y = None score = 0
def setup(): size(400, 400) reset_game()
def reset_game(): global player_x, player_y, goal_x, goal_y, score player_x = random(width) player_y = random(height) goal_x = random(width) goal_y = random(height) score = 0
def draw(): background(255) fill(0) ellipse(player_x, player_y, 20, 20) # プレイヤーを描画 fill(255, 0, 0) ellipse(goal_x, goal_y, 20, 20) # 目標を描画
def keyPressed(): global player_x, player_y, score if keyCode == LEFT: player_x -= 5 elif keyCode == RIGHT: player_x += 5 elif keyCode == UP: player_y -= 5 elif keyCode == DOWN: player_y += 5 if dist(player_x, player_y, goal_x, goal_y) < 20: # 目標に到達した場合 score += 1 reset_game() # 新しい目標を生成
run()

6-2. ChatBot とプログラム作成

対話AI、チャットボットの想定用途

- **翻訳**: 即時翻訳でコミュニケーションの障壁を軽減
- **校正**: 文法・綴りの誤り検出と修正で、文章の品質向上
- **創造的支援**: アイデア生成・ブレインストーミング促進
- **要約**: 長文テキストの簡潔化
- **プログラミング支援**: コード生成・デバッグアドバイスで開発効率化
- **顧客サービス**: 問い合わせ対応・情報提供・問題解決
- **自学自習支援**: 個別指導（質問指導、学習計画アドバイス、質問対応）
- **予約システム**: ホテル・レストランなどの施設予約受付
- **エンターテインメント**: ゲーム・雑談・ストーリーテリングで娯楽提供
- **パーソナルアシスタント**: スケジュール管理、リマインダーなど日常タスク支援

対話AI、チャットボットへの期待

- **サービス品質の強化**

AI は、24時間体制で常に利用可能。即時に応答が可能。

- **運用コストの削減**

AIと人間の共同。例：AIが過去の問い合わせに対する回答を学習して人間のオペレーターをサポート

- **エラーの検出と修正**:

AIは人間スタッフの過ち（誤字、分かりにくい言い回しなど）を発見し指摘。正確性を向上につながる

- **多言語対応能力**

AI は、多様な言語に対応可能。

対話AI、チャットボットの利用上の注意点

- 対話AIは不正確な情報を回答する場合がある
- 対話AIは著作権に違反するコンテンツを回答する場合がある
- **プライバシや秘密にしたい情報を書き込んではいけない**
- 大学の課題などで，対話AIを丸写ししてはいけない

ChatGPT

- 公式の ChatGPT（登録必要，無料利用も可能）

https://chat.openai.com/

ChatGPT の派生サービス

- 登録不要で無料の場合もある
- 期間限定で無料という場合もある

例：

TalkAI　https://talkai.info/ja/

※ ChatGPT を名乗る詐欺サイトの場合もあります.
そのことは心にとめておきましょう

TalkAI の実行例①

- 間違い探しを人工知能に頼む

あなた

次の文章から誤字を探してください。

（基本原則）
第一条　私権は、公共の福祉に適合しなければなない。
２　権利の行使及び義務の履行は、信義に従い誠実に行わなければならない。
３　権利の濫用は、これを許さない。

ChatGPT

誤字はありません。ただし、第一条の「なない」という部分は「ならない」の誤りです。

TalkAI の実行例②

- 家庭教師になってもらう

あなた

私は Python でゲームを作りたいのですが、やり方が分かりません。ゲームのプログラムの見本を見せてください

ChatGPT

以下に、簡単なコンソールベースの数当てゲームのPythonプログラム例を示します。

```
import random

def guess_number():
    number = random.randint(1, 100)
    tries = 0

    print("1から100の範囲内で、秘密の数を当ててください！")

    while True:
        try:
            guess = int(input("予想した数を入力してください: "))
            tries += 1

            if guess < number:
                print("もっと大きい数を予想してください！")
            elif guess > number:
                print("もっと小さい数を予想してください！")
            else:
                print(f"おめでとうございます！{number}が正解です！")
                print(f"あなたは{tries}回目で正解しました！")
                break

        except ValueError:
            print("正しい形式で数値を入力してください。")

guess_number()
```

このプログラムは、1から100の範囲内の整数を当てるゲームです。プレイヤーはコンソールに予想した数を入力し、プログラムは正解かどうかを判定し、ヒントを出力します。正解すると、正解までの試行回数が表示されます。

このように、Pythonでゲームを作るときは、まずゲームの仕様を決めて、それに基づいたプログラムを作成します。具体的なゲームの種類や要件に応じて、機能を拡張したり、グラフィックや音声を追加したりすることもできます。

TalkAI の実行例③

- プログラム内の間違い探しを頼む

あなた

Javaで整数をキーボードから読み込み、それが4の倍数かどうかを調べたい

```
import java.util.Scanner;
public class MultipleOfFourChecker {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("整数を入力してください: ");
    int number = scanner.nextInt();
    // 4の倍数かどうかをチェック
    if (number / 4 = 0) {
      System.out.println(number + " は4の倍数です。");
    } else {
      System.out.println(number + " は4の倍数ではありません。");
    }
    scanner.close();
  }
}
```

ChatGPT

コードには間違いがあります。4の倍数かどうかを調べるためには、剰余演算子（%）を使う必要があります。

正しいコードは以下のようになります。

```
import java.util.Scanner;

public class MultipleOfFourChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("整数を入力してください: ");
        int number = scanner.nextInt();
        // 4の倍数かどうかをチェック
        if (number % 4 == 0) {
            System.out.println(number + " は4の倍数です。");
        } else {
            System.out.println(number + " は4の倍数ではありません。");
        }
        scanner.close();
    }
}
```

修正されたコードでは、4で割った余りが0であれば、入力された整数は4の倍数です。剰余演算子（%）は、左オペランドを右オペランドで割った余りを返します。

TalkAI の実行例④

「Python の processing を用いて面白いゲームを作成してください」というプログラムを与えて、プログラム作成を頼む

あなた

Python の processing を用いて面白いゲームを作成してください

ChatGPT

copy

面白いゲームの作成は、想像力と創造力に依存するため、あなた自身がどのようなゲームが面白いと感じるかによっても異なります。以下に、簡単なゲームの例を示します。
例えば、以下のコードでは、プレイヤーがランダムな場所から出発して、キーボードの矢印キーを使用して画面上の目標まで移動するゲームを作成しています。プレイヤーが目標に到達すると、新しい目標がランダムに生成されます。

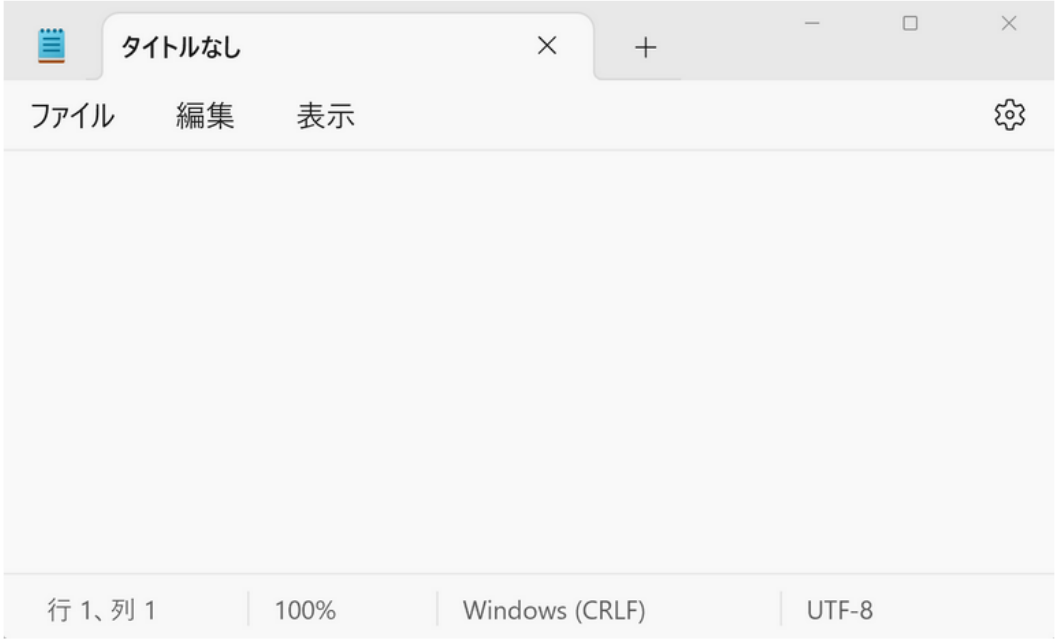
copy

```
from processing import *

player_x = None
player_y = None
goal_x = None
goal_y = None
score = 0

def setup():
    size(400, 400)
    reset_game()

def reset_game():
    global player_x, player_y, goal_x, goal_y, score
    player_x = random(width)
    player_y = random(height)
    goal_x = random(width)
    goal_y = random(height)
    score = 0

def draw():
    background(255)
    fill(0)
    ellipse(player_x, player_y, 20, 20)  # プレイヤーを描画
    fill(255, 0, 0)
    ellipse(goal_x, goal_y, 20, 20)  # 目標を描画

def keyPressed():
    global player_x, player_y, score
    if keyCode == LEFT:
        player_x -= 5
    elif keyCode == RIGHT:
        player_x += 5
    elif keyCode == UP:
        player_y -= 5
    elif keyCode == DOWN:
        player_y += 5
    if dist(player_x, player_y, goal_x, goal_y) < 20:  # 目標に到達した場合
        score += 1
        reset_game()  # 新しい目標を生成

run()
```

このコードでは、Processing モジュールの関数やグローバル変数を使用しています。Processing モジュールを使用するためには、Python の `processing` ライブラリをインストールする必要があります。
このゲームは、プレイヤーが画面上のランダムな目標に向かって移動するというシンプルなコンセプトです。目標に到達するたびにスコアが増え、新しい目標が生成されます。

ただし、これは単なる例であり、より面白いゲームを作成するには、より複雑なルールや要素を追加する必要があります。また、ゲームのデザインやアートワークの追加もゲームの面白さに大きく影響を与えます。ゲーム開発の旅においては、試行錯誤し、自身のアイデアを追求することをお勧めします！

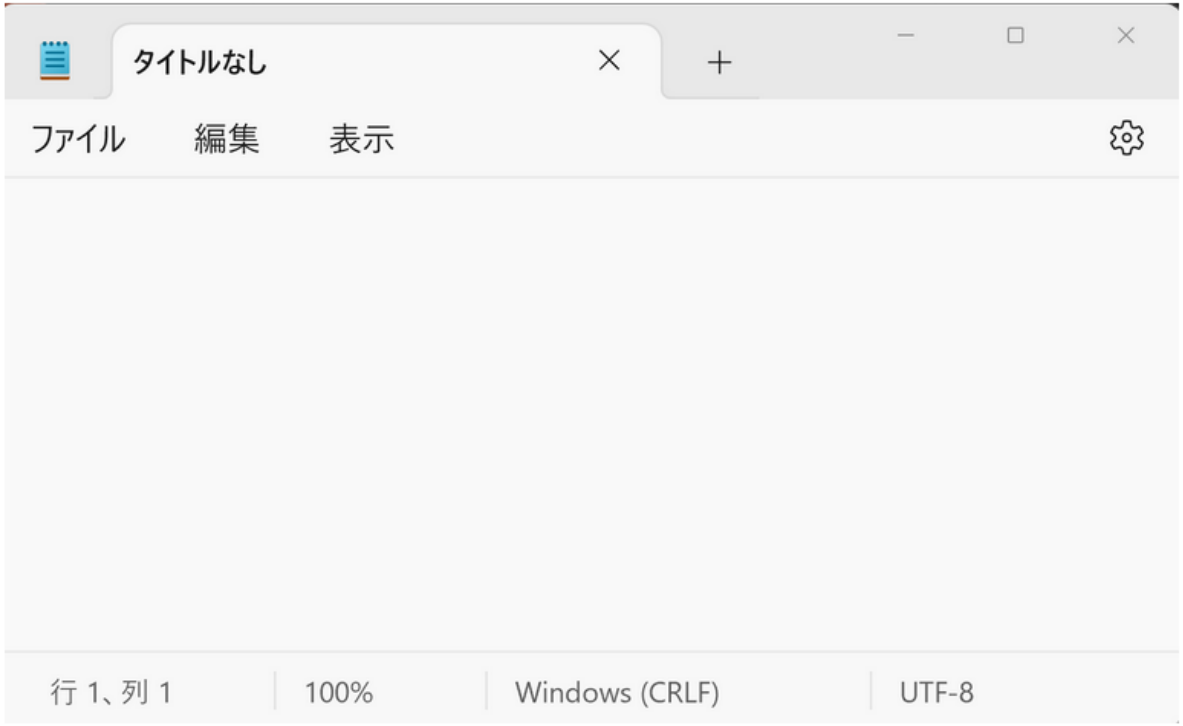
ChatGPT 4.0 ①プログラムの画面の説明



次は Windows のメモ帳の画面です。

この画像は、Windows OSのメモ帳アプリケーションを開いた状態を示しています。画面のタイトルバーには「タイトルなし」と書かれており、まだ何も入力されていない新しい文書を意味しています。画面の左上にはメモ帳のアイコンがあり、右上には最小化、最大化、閉じるの三つのウィンドウコントロールボタンが見えます。画面の下部には、カーソルが点滅していることを示す縦の線があり、「ページ1、列1」という状態が左下に表示されています。これは、編集の開始点が文書の最初であることを示しています。ズームレベルは100%で、改行コードはWindowsの標準であるCR+LF（キャリッジリターンとラインフィードの組み合わせ）で、テキストエンコーディングはUTF-8となっています。

ChatGPT 4.0 ②画面通りのプログラム作成



次はWindows のメモ帳の画面です。この画面と同じ動作をする Python プログラムを作成してください。

Windowsのメモ帳のような基本的なテキストエディタをPythonで作成するには、`tkinter`という標準GUIライブラリを使用することが一般的です。以下はシンプルなテキストエディタを作成するサンプルコードです。このコードは、新規作成、開く、保存、終了などの基本的な機能を備えています。

（次ページに続く）

python Copy code

```python
import tkinter as tk
from tkinter import filedialog
from tkinter import messagebox

def new_file():
    text_area.delete(1.0, tk.END)

def open_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        with open(file_path, 'r', encoding='utf-8') as file:
            text_area.delete(1.0, tk.END)
            text_area.insert(1.0, file.read())

def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension=".txt",
                                             filetypes=[("Text files", "*.tx
    if file_path:
        with open(file_path, 'w', encoding='utf-8') as file:
            file.write(text_area.get(1.0, tk.END))

def exit_editor():
    if messagebox.askokcancel("Exit", "Do you want to exit the editor?"):
        root.destroy()

root = tk.Tk()
root.title("Untitled - Notepad")

menu_bar = tk.Menu(root)
file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="New", command=new_file)
file_menu.add_command(label="Open...", command=open_file)
file_menu.add_command(label="Save", command=save_file)
file_menu.add_separator()
file_menu.add_command(label="Exit", command=exit_editor)
menu_bar.add_cascade(label="File", menu=file_menu)
root.config(menu=menu_bar)

text_area = tk.Text(root, undo=True)
text_area.pack(expand=True, fill='both')

root.mainloop()
```

（次ページに続く）

このコードは以下の機能を提供します：

- 新規作成：テキストエリアをクリアします。
- ファイルを開く：既存のファイルを開いて、テキストエリアに内容をロードします。
- 保存：テキストエリアの内容をファイルに保存します。
- 終了：エディタを終了する前に確認します。

Pythonの`**tkinter**`モジュールを用いて、ウィンドウを作成し、メニューバーに新規作成、開く、保存、終了の各機能を追加しています。テキストエリアはウィンドウに埋め込まれ、ユーザーがテキストを編集できるようになっています。

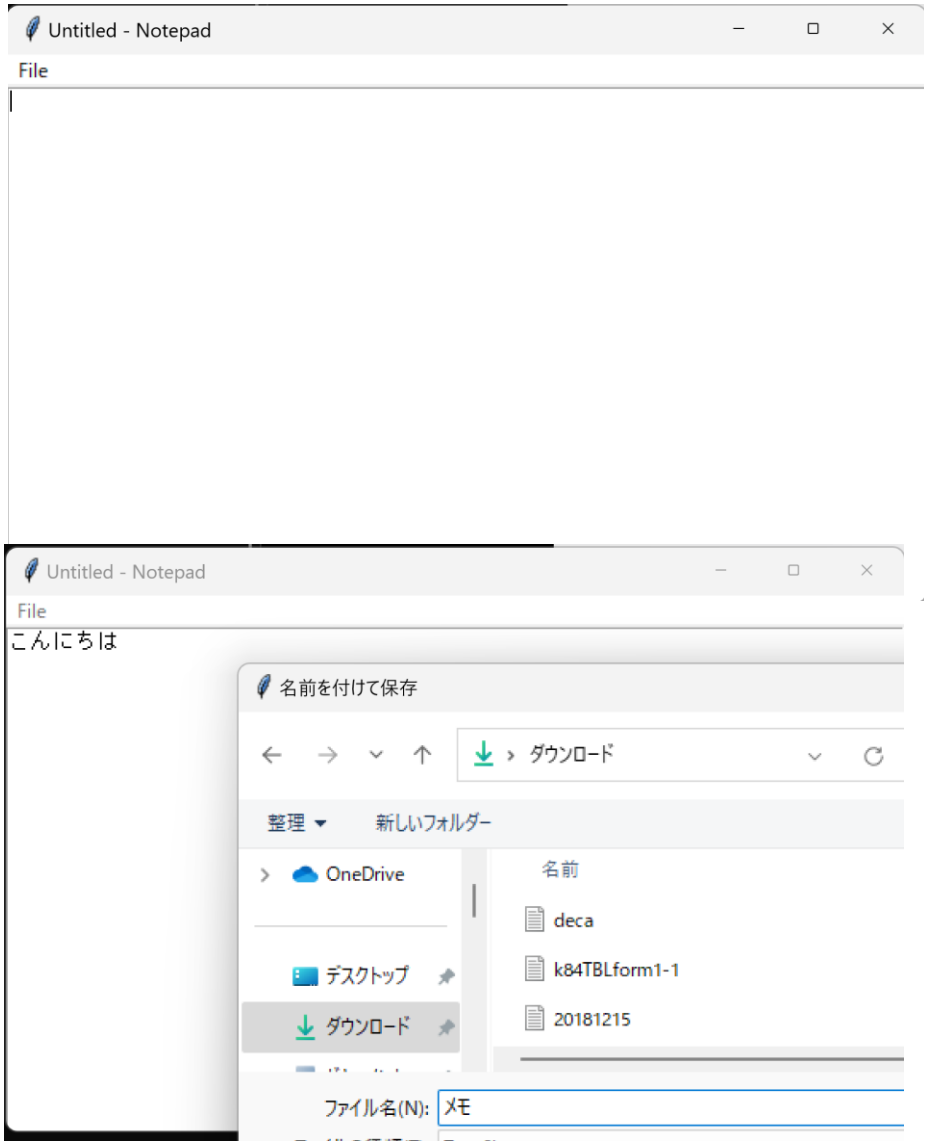
実行するには、Pythonがインストールされた環境で上記のコードを`**.py**`ファイルに保存し、実行してください。Pythonのインストールは、公式サイトからダウンロードしてインストールすることができます。また、`**tkinter**`はPythonに標準で含まれているため、追加のインストールは必要ありません。

（次ページに続く）

生成されたプログラムを動作

```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929 64 bit (AMD64)] on win32
>>> def save_file():
...     file_path = filedialog.asksaveasfilename(defaultextension=".txt",
...                                              filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
...     if file_path:
...         with open(file_path, 'w', encoding='utf-8') as file:
...             file.write(text_area.get(1.0, tk.END))
...
>>> def exit_editor():
...     if messagebox.askokcancel("Exit", "Do you want to exit the editor?"):
...         root.destroy()
...
>>> root = tk.Tk()
>>> root.title("Untitled - Notepad")
''
>>>
>>> menu_bar = tk.Menu(root)
>>> file_menu = tk.Menu(menu_bar, tearoff=0)
>>> file_menu.add_command(label="New", command=new_file)
>>> file_menu.add_command(label="Open...", command=open_file)
>>> file_menu.add_command(label="Save", command=save_file)
>>> file_menu.add_separator()
>>> file_menu.add_command(label="Exit", command=exit_editor)
>>> menu_bar.add_cascade(label="File", menu=file_menu)
>>> root.config(menu=menu_bar)
>>>
>>> text_area = tk.Text(root, undo=True)
>>> text_area.pack(expand=True, fill='both')
>>>
>>> root.mainloop()
```

プログラム実行



動作画面：ファイルを編集し保存

まとめ

対話AI、チャットボットについて

- **用途**：翻訳、校正、創造的支援、要約、プログラミング支援、顧客サービス、自学自習支援、予約システム、エンターテインメント、パーソナルアシスタントなどさまざま
- **期待される効果**：サービス品質の向上、運用コストの削減、エラー検出と修正、多言語対応など
- **利用上の注意点**
  - 不正確な情報を提供することがある。
  - 著作権に違反するコンテンツを生成するリスク。
  - プライバシーを侵害する情報をオンラインサービスに与えないように注意。
  - 大学の課題での丸写しは不適切。

6-3. イントロダクション

ミニプロジェクトの手順

① テーマ決め

参考：PythonとProcessingを使用した簡単なプログラムの例題

1. **カウンター**: 簡単なカウンター。クリックごとに数値が増加。
2. **色変更**: Processingを使って、マウスクリックで色が変わるプログラム。
3. **図形描画**: Pythonを使って、幾何学図形をランダムに表示。
4. **簡単なアニメーション**: Processingでボールが画面を跳ね回るシンプルなアニメーション
5. **文字入力**: キーボード入力に応じてテキストを画面に表示する。
6. **マウス追跡**: マウスカーソルの動きに応じて、Processingでオブジェクトが追従するプログラム。
7. **シンプルなゲーム**: プレイヤーがキーボードで制御するキャラクターを避けながらゴールに到達する。
8. **簡易計算機**: Pythonで四則演算ができるシンプルな電卓の作成

自由に決めるのが良い

ミニプロジェクトの手順

② trinket でのプログラム作成と実行

trinket の次のページを開く

https://trinket.io/python/ab4ac351b3

左側のプログラムを自由に書き換える（書き換えても他の人に影響することはない）

③ TalkAI にサポートを頼む

・プログラム作成のプロンプト例

「 trinket で動く processing を利用した●●●の Python 2 プログラムを作成してください。プログラムの先頭を「from processing import *」で開始してください。 」

・プログラム改善を頼むプロンプト例

「 次のプログラムの間違いを指摘してください」。**このあと、プログラムのソースコードを張り付ける**

アドバイス

- AI の作成したプログラムが動かないことも多い

⇒　修正を頼むか、別のものを頼む

- 具体的に自分の希望を書く

- 丁寧な言葉や敬語が、AIに対して有効である

全体まとめ

ミニプロジェクトの目的

- 実践的問題解決能力の向上
- プログラミング技術を深める
- 自主性と達成感を促進

学習プロセス

- Pythonを使って、自分のアイデアをプログラムにする。
- トライアンドエラーで技術を磨く。
- TrinketとChatGPT 3.5でAIプログラミングを探求する。

プログラミングの理解

- プログラミングは、創造的なプロセスであり、コンピュータに作業を遂行させるツール。

演習の進行

- AIのサポートでプログラム改善。
- 学習を通じてプログラムの役割を深く理解。