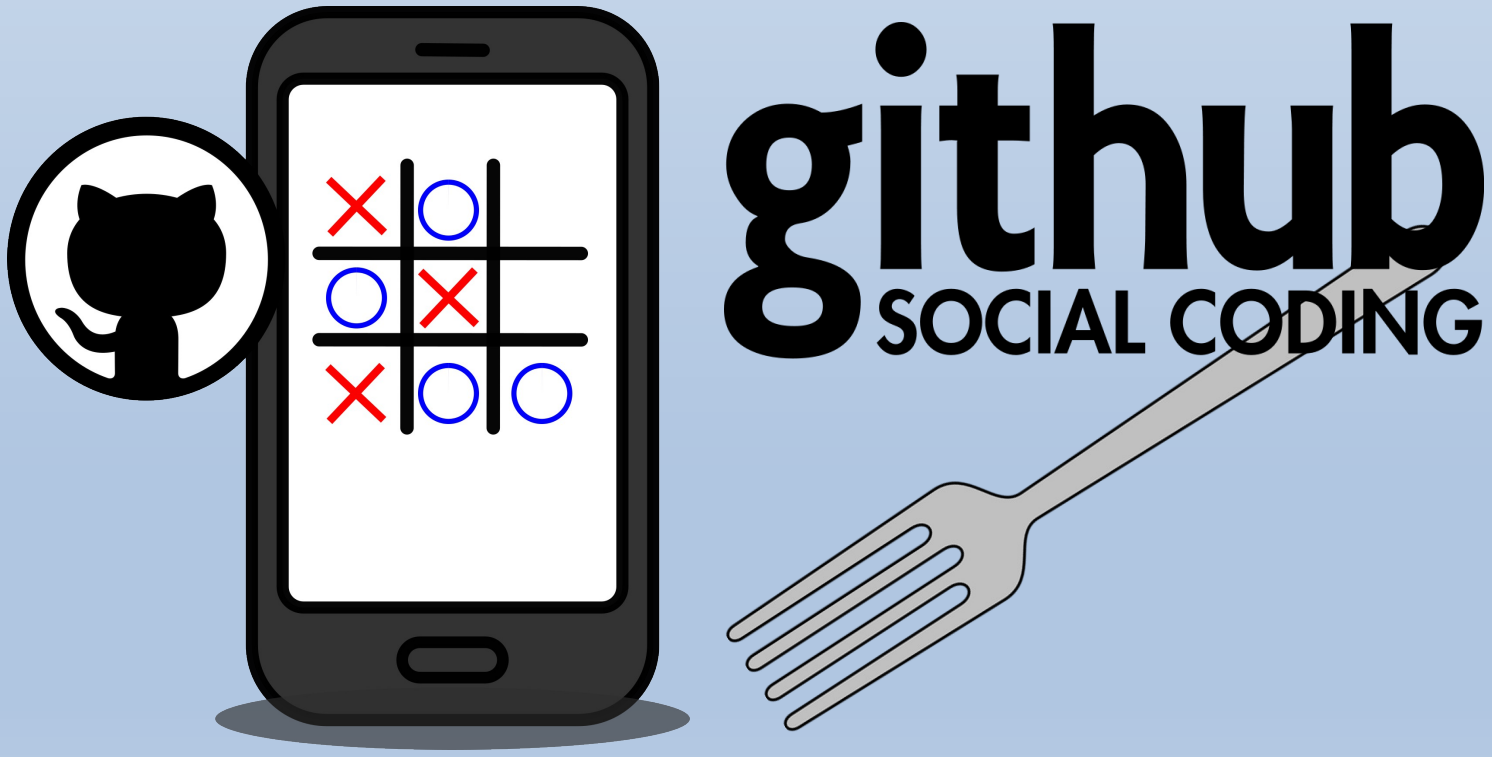
SCIENCE, TECHNOLOGY, ENGINEERING & MATHEMATICS (STEM)
OCTOBER 2025

By Eden Lattibeaudiere (BA, MSc)

Introduction to Artificial Intelligence, Python, GitHub and Copilot

Develop a two player Python based Tic-Tac-Toe game with an optional AI opponent upgrade and distribute it via GitHub



https://github.com/skelphx/thecloud.git

Table of Contents

Chapter 1

Introduction

This tutorial offers a gentle introduction to the world of Python Programming, Artificial Intelligence, GitHub and Copilot. It is suitable for beginners, intermediate level enthusiasts and advanced practitioners alike. Have Fun!

You'll build:

(i) A complete two player game.

(ii) A version with an **AI Opponent.**

(iii) A project you can upload and showcase to the world via **GitHub.**

Chapter 2.0

What is Python?

Introduction to Python

Python is an open-source, cross platform programming language that has become increasingly popular over the past ten years. It was first released in 1991 and the latest version is 3.13.7. CPython is the reference implementation of the Python Programming Language written in C. CPython is the default and most widely used implementation of the Python.

Python is also multi-purpose (due to it's many versatile extensions). Examples of Python extensions and how they are used include various implementations in scientific computing and calculations, simulations, web development, artificial intelligence and gaming. Python is now considered the most popular programming language in the world today.

Chapter 2.1

Python Basics

Concepts Covered:

(i) Variables.

(ii) Lists and Loops.

(iii) Functions.

(iv) Classes.

(v) Input/Output.

(vi) Simple AI logic.

(vii) GitHub basics.

Chapter 2.2

What is GitHub?

Introduction to GitHub

GitHub is a cloud-based platform that enables developers to store, manage, and collaborate on code using Git, a version control system created by Linus Torvalds, who also happens to be the creator of the Linux operating system. GitHub makes it easier for teams of all sizes to collaborate on projects by providing tools for tracking changes, reviewing code, versioning software and integrating countless other development tools and services within the platform.
Whether you're working solo or as part of a large open-source project, GitHub helps you keep your code organized and your workflow efficient. GitHub is also free to use.

Chapter 3

Game Structure

You'll build the game using a Python class called **TicTacToe** and save class TicTacToe and its functions in a script called **tictactoe.py**. The file extension **.py** denotes it is a Python script specifically for use with the Python Interpreter.

It handles:

(i) Game state (board, turn, winner).

(ii) Display logic**.**

(iii) Functions**.**

(iv) Player input.

(v) Win/draw checking.

Chapter 4.0

class TicTacToe

When reviewing the Tic-Tac-Toe script be aware that the Python keyword **class** creates a class and the keyword **def** creates a function.

A Python class is similar in structure to the chapter of a book, where the book itself is the script. A function, metaphorically speaking, could be described as the vocabulary used present a particular concept to the reader in order to help them fulfill a specific purpose or requirement as a direct result of reading the book. Each function is contained within a gray block for presentation purposes. Functions are preceded by a commented line instantiated with a hash tag "**#**" and a number identifying its position in **class TicTacToe** then proceeded by a brief explanation of its syntax.

The Tic-Tac-Toe script is comprised of a single, solitary class:

class TicTacToe

And nine functions:

__init__() , display_board(), is_valid_move(), make_move(), check_winner(), switch_player(), get_move(), play() and main().

Chapter 4.1

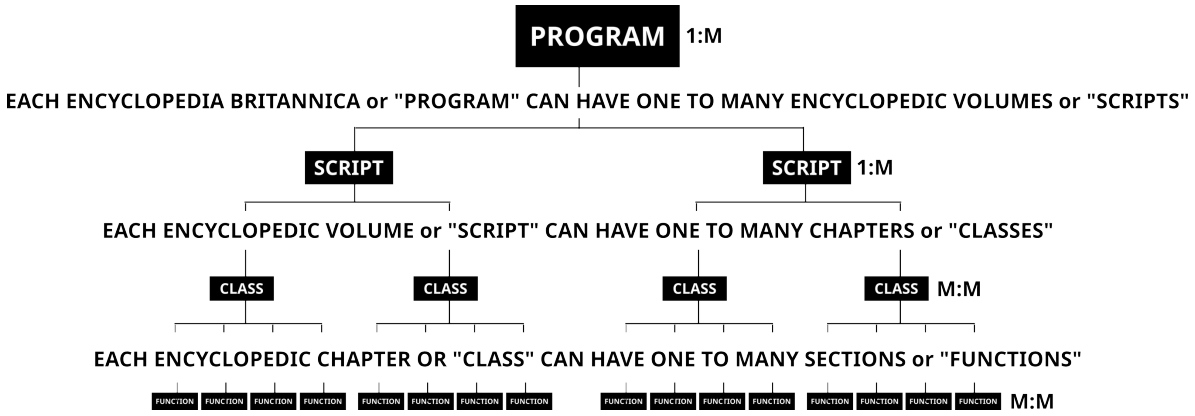
Program Construct (PP vs OOPs)

"PROGRAM" ENCYCLOPEDIA BRITANNICA

OBJECT REFERENTIAL/ RELATIONAL INTERGRITY CONSTRAINT NOTATION

1:M = one-to-many relationship

M:M = many-to-many relationship



EACH ENCYCLOPEDIC SECTION or "FUNCTION" could be comprised of "SPECIFIC INFORMATION, DATA, GRAMMAR, PUNCTUATION etc"

The "SPECIFIC INFORMATION, DATA, GRAMMAR, PUNCTUATION etc" could be interpreted as the Keywords, Statements, Conditional Statements, Operators, Operands, Variables, Arrays etc normally found in a computer program.

Scripts, Classes, Functions and certain types of Variables and Arrays (known as "Globals") are often reuseable and called from within other Programs, Scripts, Classes and Functions. Under these circumstances Scripts, Classes, Functions etc are classified as Objects. Objects are used in Object Orientated Programming (OOP). Object Orientated Programming is not the same as Procedural Programming but can contain Objects found in Procedural Programs.

Chapter 5.0

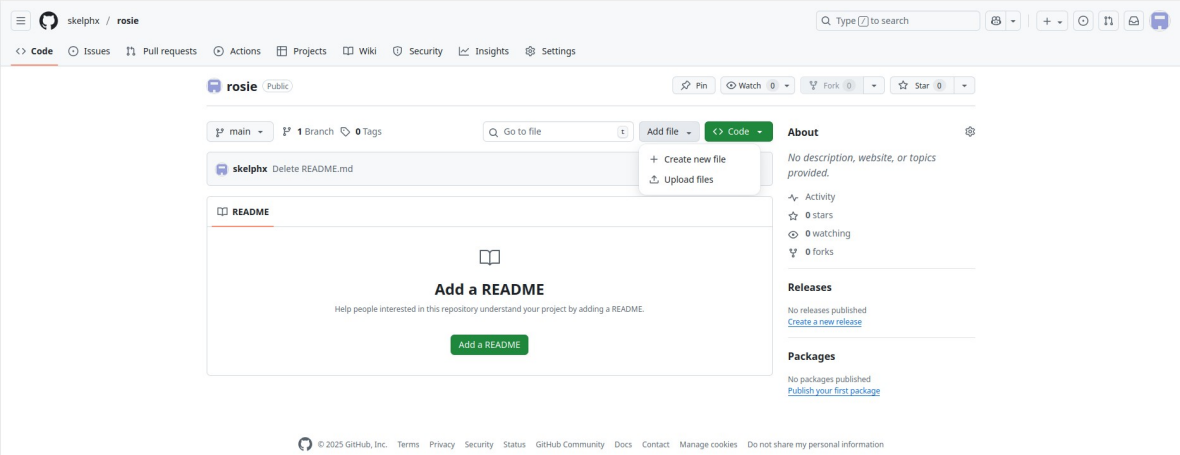
tictactoe.py

Follow each step in this tutorial to create **tictactoe.py.**

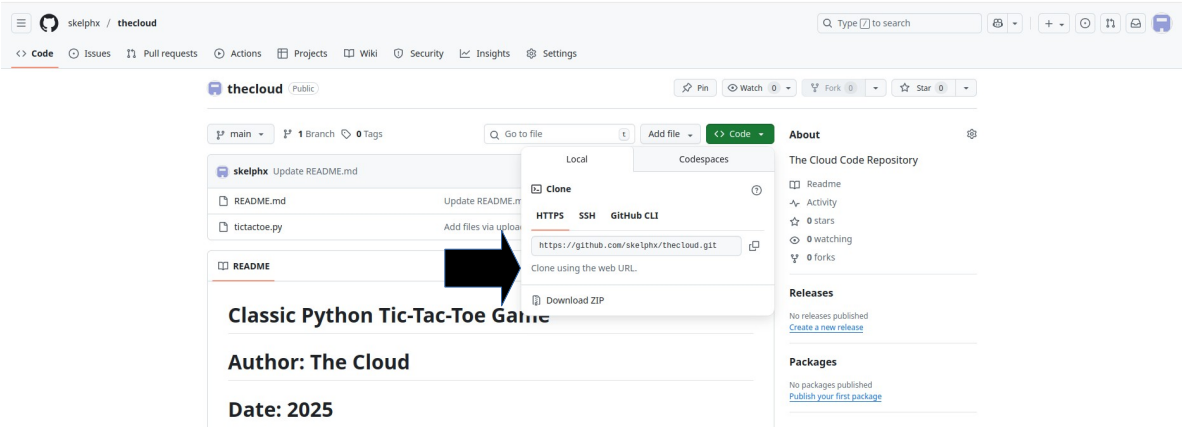
(i) Review **Chapter 9** and create a **GitHub Repository** etc.

(ii) Click **Add File** then **Create new file** (see screenshot below).

(iii) Enter the code snippets featured on the following pages, name the file **tictactoe.py** then press **Commit changes**.



(iv) Alternatively you can download the entire tic-tac-toe zip archive from **https://github.com/skelphx/thecloud.git**.



Chapter 5.1

Tic-Tac-Toe

Note: Do not include line numbers, these are just placeholders. Only enter the pseudo code with the gray background and repeat for each additional block in the sequence. This is known as procedural programming: *Block 1, Block 2, Block 3 … etc.*

Block 1

```
1   #Import os and sys Standard Python Library Modules
2
3   import os
4   import sys
5
6   #Initialize class TicTacToe
7
8   class TicTacToe:
9
10   #1  __init__()  allocates nine blank spaces to self.board variable, sets current player to X and
11   starts a new game where the winner is nullified.
12
13  def __init__(self):
14          self.board = [' '] * 9
15          self.current_player = 'X'
16          self.game_over = False
17          self.winner = None
```

Block 2

```
1   #2 display_board() clears screen, prints title, columns separated by bars, rows separated by hyphenated separators
2    of the same length.
3
4   def display_board(self) :
5      os.system('cls' if os.name == 'nt' else ' clear')
6      print("\nTic-Tac-Toe\n")
7      for i in range(3):
8         row = ""
9         for j in range(3):
10            pos = i * 3 + j
11            row += f" {self. Board [pos]} "
12            if j < 2:
13               row += "|"
14         print(row)
15         if i < 2:
16            print("------------")
17      print()
```

Chapter 5.2

Block 3

```
#3 is_valid_move() if position is empty and between >=0 & <=9 it is a valid position.

def is_valid_move(self, position):
try:
pos = int(position) - 1
return 0 <= pos <= 8 and self.board[pos] == ' '
except ValueError:
return False
```

Block 4

```
#4 make_move -1 from positions and play.

def make_move(self, position):
pos = int(position) - 1
self.board[pos] = self.current_player
```

Block 5

```
#5 check_winner() after move check winning combination and that all board spaces are not ' ', if check proves
true and there is a winner or a draw, end game is True, if not end game if False.

def check_winner(self):
    combos = [
        [0,1,2],[3,4,5],[6,7,8],
        [0,3,6],[1,4,7],[2,5,8],
        [0,4,8],[2,4,6]
    ]
      for combo in combos:
          if self.board[combo[0]] == self.board[combo[1]] == self.board[combo[2]] != ' ':
              self.winner = self.board[combo[0]]
              self.game_over = True
              return True
          if ' ' not in self.board:
             self.winner = 'Draw'
             self.game_over = True
         return True
         return False
```

Block 6

```
#6 switch_player() switches to other player.

def switch_player(self):
self.current_player = 'O' if self.current_player == 'X' else 'X'
```

Chapter 5.3

Block 7

```
#7 get_move() allow move to positions 1-9 if 0 positions, else if move is to invalid
position display error.

    def get_move(self):
    while True:
        move = input(f"Player {self.current_player}, choose (1–9): ")
          if move.lower() in ['q', 'quit', 'exit']:
                        sys.exit("Thanks for playing!")
          if self.is_valid_move(move):
                        return move
                        print("Invalid move. Try again.")
```

Block 8

```
#8 play() starts game then only when  playing, call display_board(), get_move(),
make_move(), check_winner(), switch_player and display_board() again, if
check_winner() indicates a winner, exit, print and concatenate Result: with
winning player.

def play(self):
    input("Press Enter to begin...")
    while not self.game_over:
        self.display_board()
          move = self.get_move()
          self.make_move(move)
          if self.check_winner():
              break
          self.switch_player()
      self.display_board()
      print(f" Result: {self.winner}!")
```

Block 9

```
#9 main() if game is won, decide whether or not to play again.

def main():
          while True:
           game = TicTacToe()
           game.play()
             again = input("Play again? (y/n): ").lower()
           if again != 'y':
           Break
          if __name__ == "__main__":
          main()
```

Chapter 6

Invoke Python Interpreter

The easiest way to invoke the Python Interpreter in any modern version of the Windows Operating System is to click **Windows Start** then in the search bar type **CMD** (Windows Command Prompt).

Click **Run as Administrator**.

Download ZIP archive (Chapter 5) from your GitHub Repository to a folder on your Windows computer and extract. Type **CD \directory** (substitute **directory** with the folder name where the archive was extracted to).

Again using **CMD**, execute the following command:

python tictactoe.py

The Python Interpreter should run the script and display the Tic-Tac-Toe game screen as seen below. If the game does not start and the Interpreter returns error(s), check Python is correctly installed, the path variable is set, then examine the Python script for syntax errors.

Before playing, review Chapter 8.4 - Tic-Tac-Toe Strategy Tips.

Chapter 7

get_move() - AI Opponent Upgrade

Compare the differences between each **get_move()** function to gain a more concise understanding of programming constructs, syntax, keywords, operators and statements. The **get_move()** upgrade featured below is optional.

Block 7

```
def get_move(self):
if self.current_player == 'X':
    while True:
        move = input("Your move (1–9): ")
        if move.lower() in ['q', 'quit', 'exit']:
            sys.exit("Game exited.")
        if self.is_valid_move(move):
            return move
        print("Invalid. Try again.")
  else:
      available = [str(i+1) for i, val in enumerate(self.board) if val == ' ']
     move = random.choice(available)
      print(f"AI chooses: {move}")
      return move
```

Create a new file in your GitHub repository see **Chapter 5**.

Enter the code snippets in **Chapters 5.1, 5.2 and 5.3**.

Replace the **get_move()** function in **Chapter 5.3, Block 7** with the **get_move()** function upgrade featured above in the gray background. Do not include the line numbers**.** Add **import random** underneath import os and import sys in the script header. Save the file as **tictactoeai.py** then proceed to **Chapter 6.** Invoke the Python Interpreter and execute **tictactoeai.py.** Bug test until **tictactoeai.py** compiles correctly without errors and the Tic-Tac-Toe AI game screen is displayed. Can you beat the AI Opponent?

Chapter 8

Tic-Tac-Toe Strategy Tips

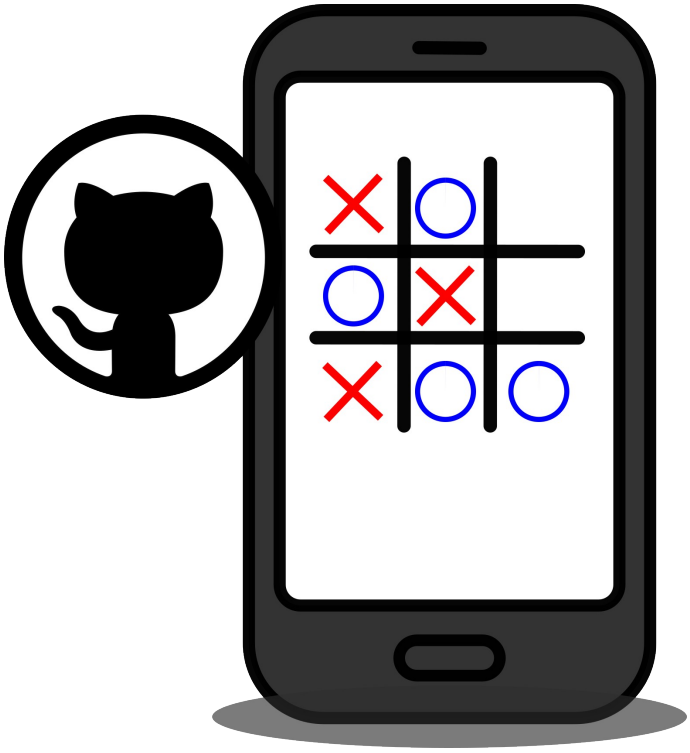
Rule 1: Control the center square.

Rule 2: Block the opponents winning move.

Rule 3: Corners are better then edges,

Rule 4: Avoid giving Forks to an opponent.

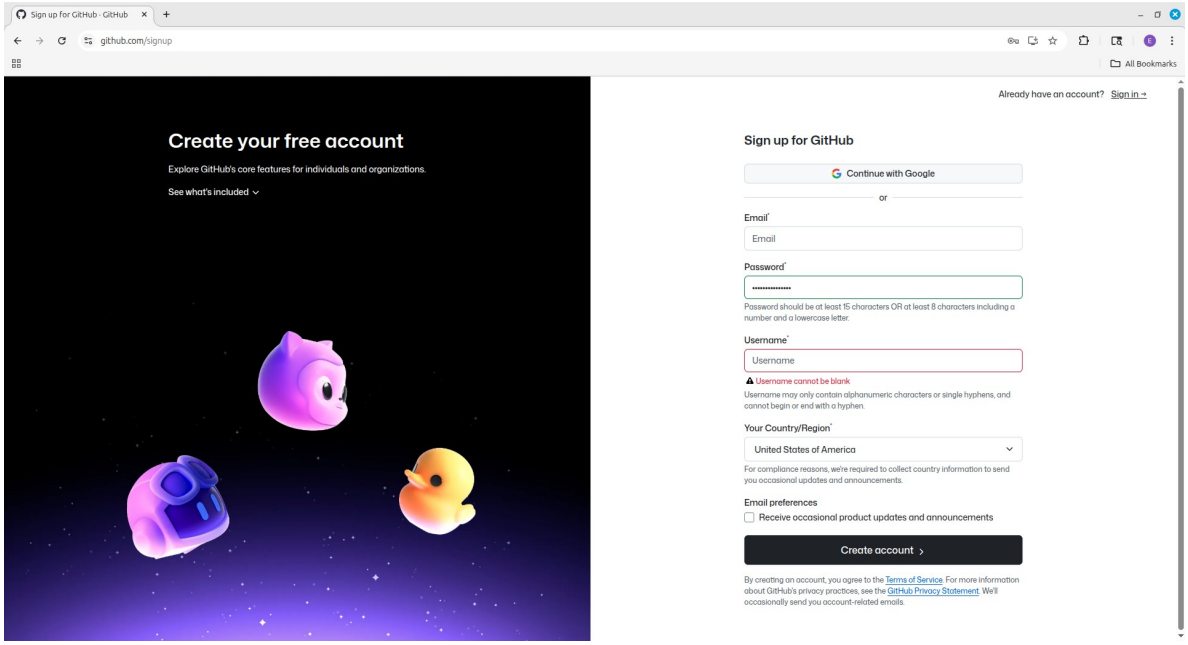
Rule 5: There is no Spoon (Matrix joke lol).



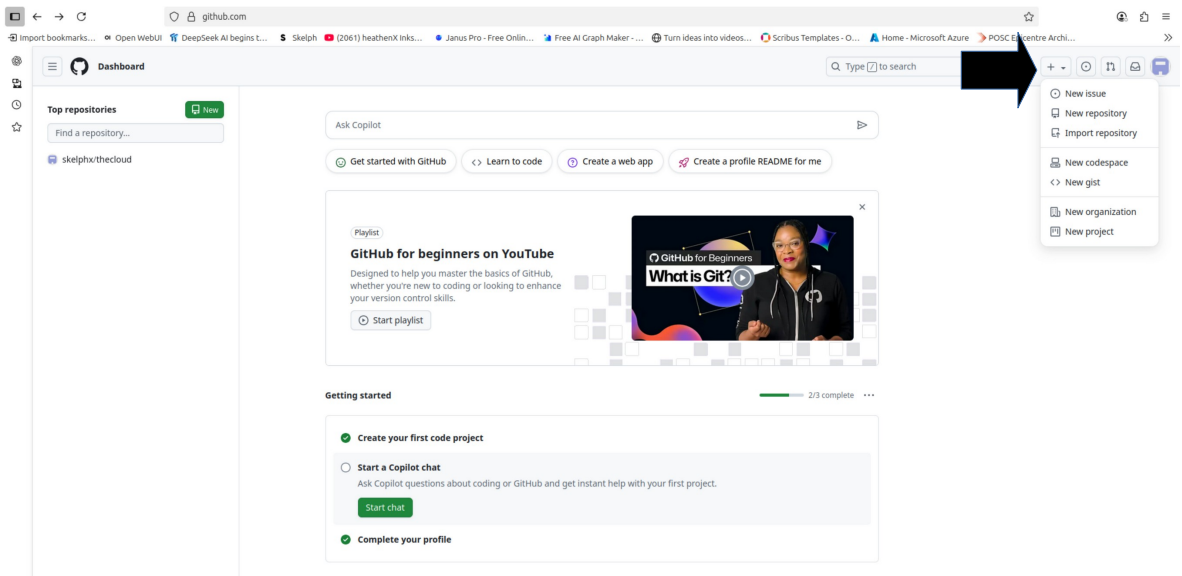
Chapter 9.0

GitHub Account / Repository Setup

(i) Open your browser and type **https://github.com/signup** in the URL bar. Enter your credentials in the Create GitHub Account Page and click **Create Account**.



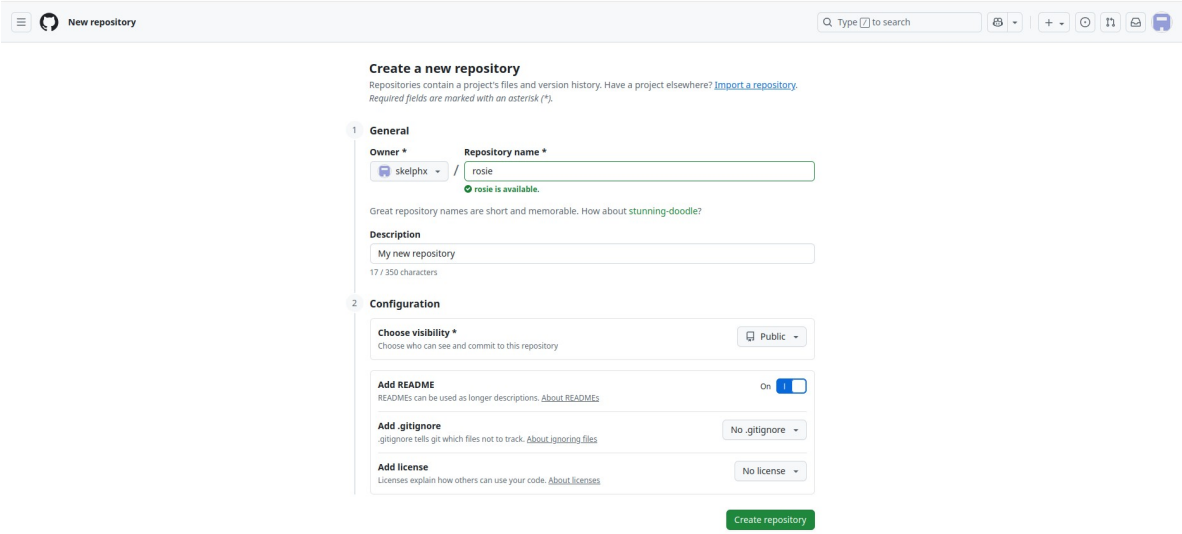
(ii) Click "**+**" on GitHub Main Menu and select **New Repository** from the drop down.



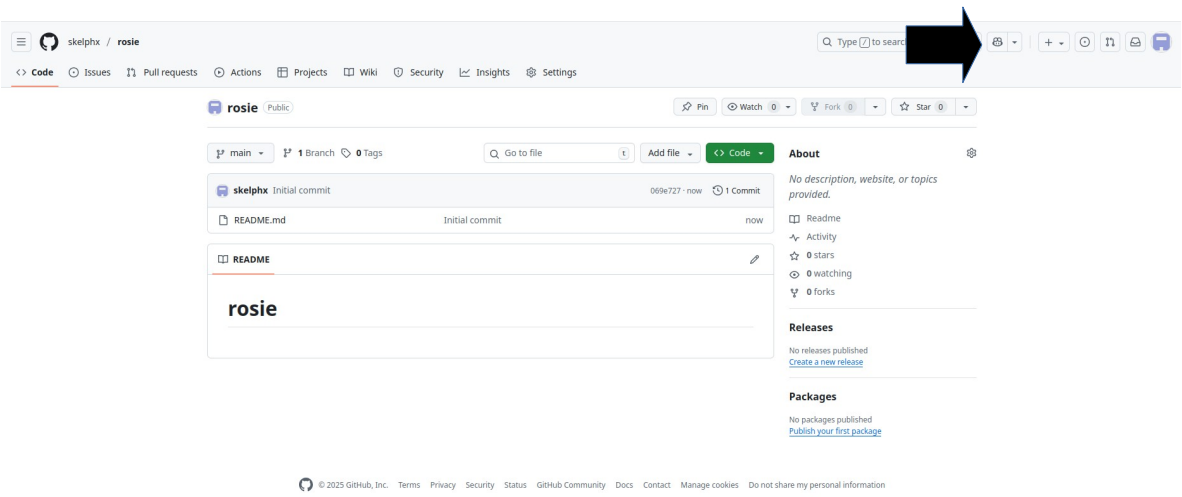
Chapter 9.1

GitHub Repository Configuration

(iii) Select **Public** (see example below) ensure **Add README** is set to **On**, then click the green **Create Repository** button.



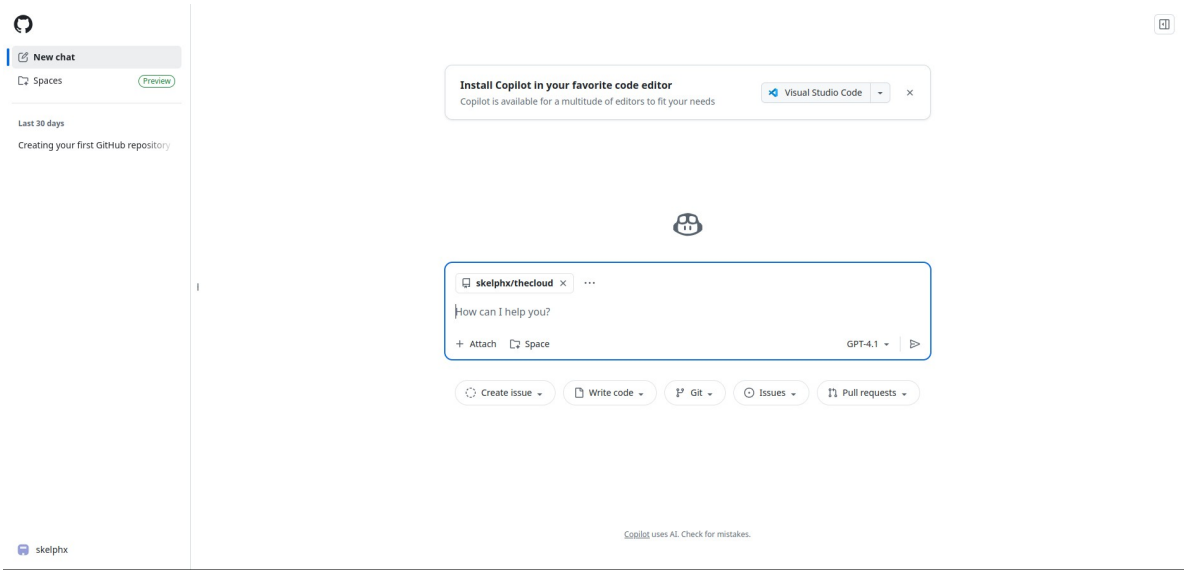
(iv) **Great Job!** – You have successfully created a new GitHub account with public repository where you can share your projects over the Internet. You can access **GitHub Copilot** by clicking the **GitHub Copilot icon** (see below).



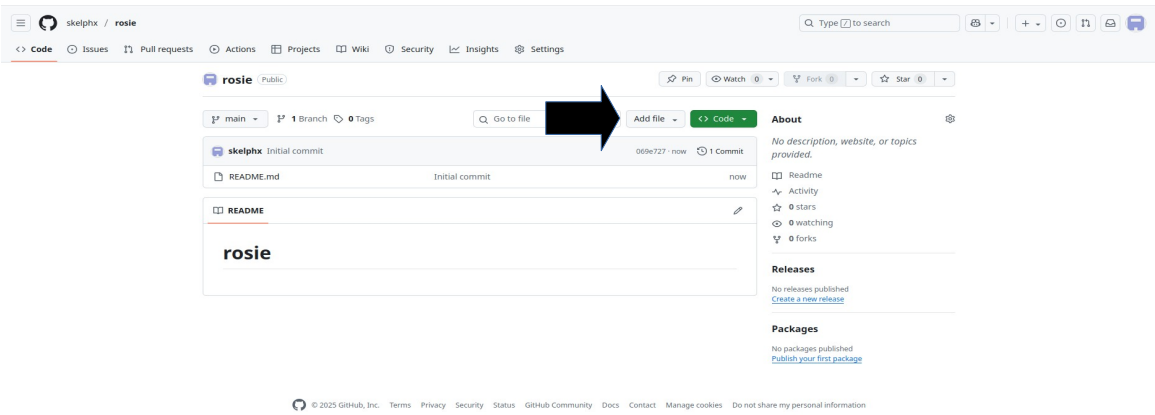
Chapter 9.2

CoPilot / GitHub Repo Add File

(v) GitHub Copilot can help you with your GitHub projects.



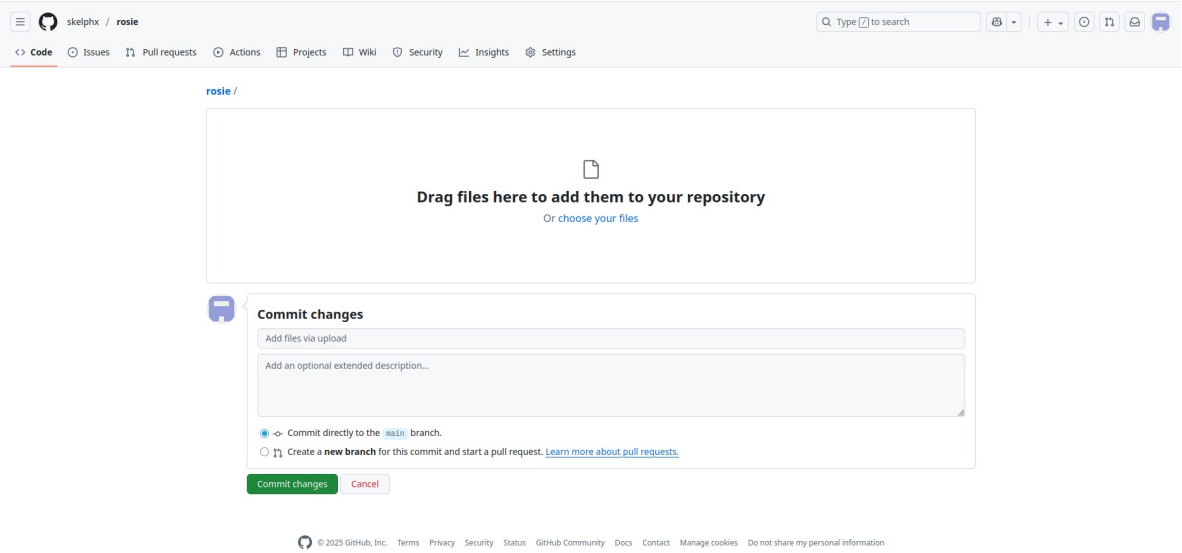
(vi) Click **Add File** from you Repo (see screenshot below) to upload the **tictactoe.py** file you downloaded in **Chapter 5**. You should see the **Add File** dialogue window in **Chapter 9.3**. Files are uploaded to your GitHub Repository from here. This step is unnecessary if you have already created **tictactoe.py** in **Chapter 5**. To help prevent overwriting important files, review the contents of your **GitHub Repository** prior to uploading anything new.



Chapter 9.3

GitHub Repo Add File Window

(iii) This is the page you access to upload files to your GitHub Repository.



Chapter 9.4

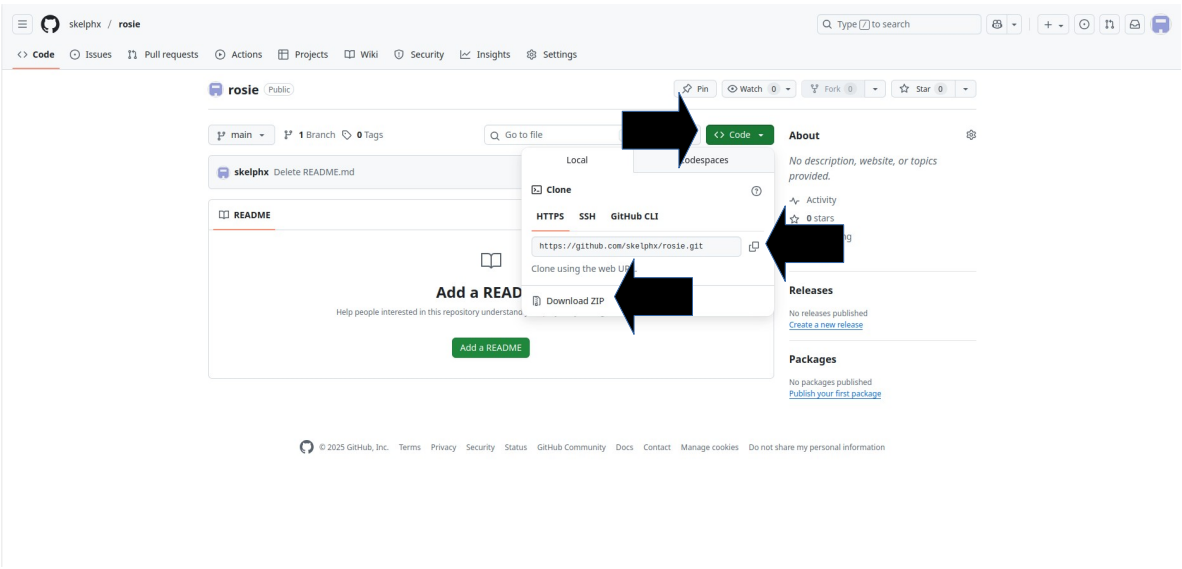
GitHub Repository Distribution

To distribute your Tic-Tac-Toe game or any other project in your GitHub Repository, click the **green code button** then the **copy button** to the right of the URL displayed in the drop-down window (see screenshot).

Your Repository will have a slightly different URL than the one displayed in the example below.

Paste the URL directly into emails, documents, text messages etc. Clicking the link or copying it into a browser URL bar will open your GitHub Repository page.

The ZIP archive containing the project files in your Repository can be downloaded by clicking **Download ZIP** (see screenshot).



Chapter 10

Summary

If you have completed the tutorial – Well Done!

You have covered A LOT.

(i) Basic Python Programming Syntax and Terminology.

(ii) Develop a 2 player Tic-Tac-Toe game with an AI Opponent upgrade using the Python programming language.

(iii) How to invoke the Python Interpreter and execute a Python script.

(iii) Configure a GitHub Account and Repository.

(iv) Create and Edit a File in GitHub.

(v) Upload Files to a GitHub Repository.

(vi) Access GitHub Copilot (AI Assistant).

(vii) GitHub Repository Distribution.

Chapter 10

Summary

If you have completed the tutorial – Well Done!

You have covered A LOT.

(i) Basic Python Programming Syntax and Terminology.

(ii) Develop a 2 player Tic-Tac-Toe game with an AI Opponent upgrade using the Python programming language.

(iii) How to invoke the Python Interpreter and execute a Python script.

(iii) Configure a GitHub Account and Repository.

(iv) Create and Edit a File in GitHub.

(v) Upload Files to a GitHub Repository.

(vi) Access GitHub Copilot (AI Assistant).

(vii) GitHub Repository Distribution.