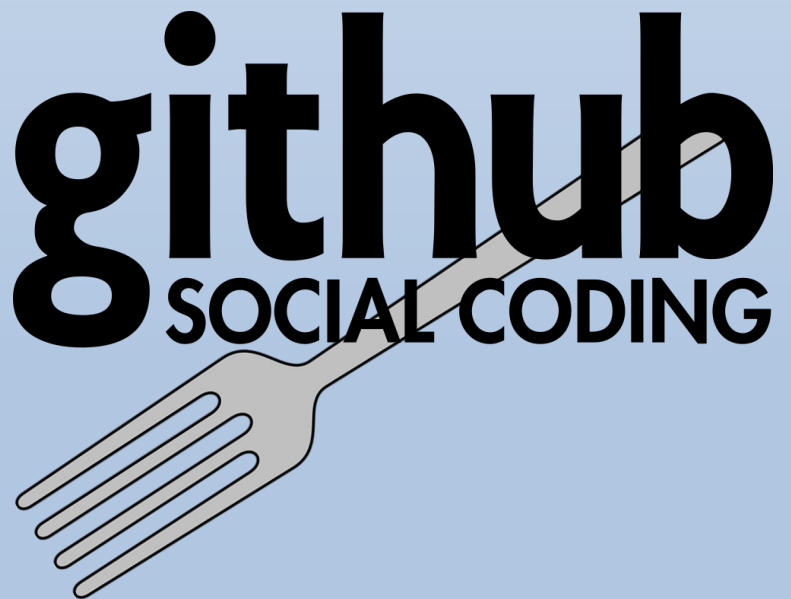
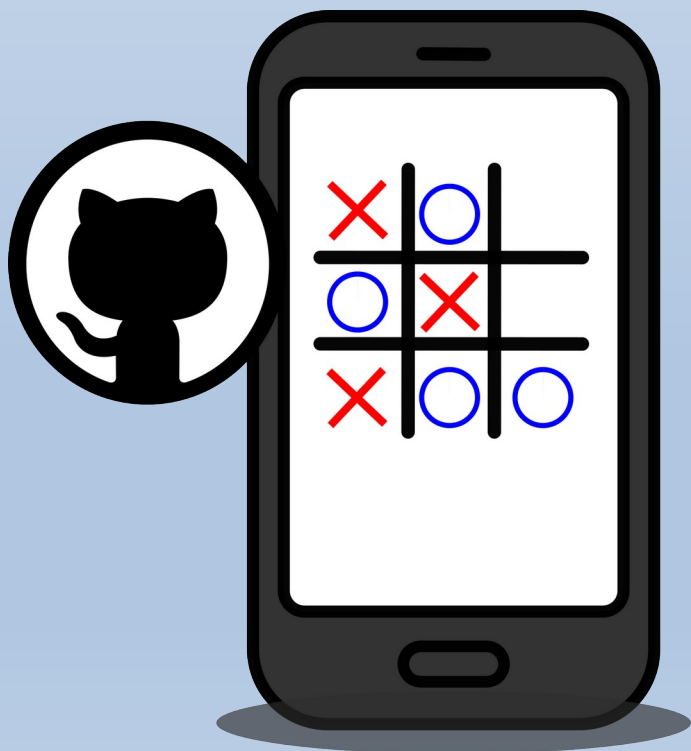


# Mastering Cloud Essentials

**Course Title: An introduction to Python and GitHub**

**Episode One: Develop a two player Python based Tic-Tac-Toe game with an optional AI opponent upgrade and share it on GitHub**



# Table of Contents

<b>Chapter 1 - Introduction</b>	<b>Page 2</b>
<b>Chapter 2 – What is Python?</b>	<b>Page 3</b>
<b>Chapter 2.1 – Python Basics</b>	<b>Page 4</b>
<b>Chapter 2.2 – What is GitHub?</b>	<b>Page 5</b>
<b>Chapter 3 – Game Structure</b>	<b>Page 6</b>
<b>Chapter 4 – class TicTacToe</b>	<b>Page 7</b>
<b>Chapter 5 – tictactoe.py - Create/Edit File GitHub Repository</b>	<b>Page 8</b>
<b>Chapter 5.1 – import, function <code>_init()</code>, <code>display_board()</code></b>	<b>Page 9</b>
<b>Chapter 5.2 – <code>is_valid_move()</code>, <code>make_move()</code>, <code>check_winner()</code>, <code>switch_player()</code></b>	<b>Page 10</b>
<b>Chapter 5.3 – <code>main()</code>, <code>get_move()</code>, <code>play_self()</code></b>	<b>Page 11</b>
<b>Chapter 6 – Invoke Python Interpreter (Windows)</b>	<b>Page 12</b>
<b>Chapter 7 – function <code>get_move()</code> - AI Opponent Upgrade</b>	<b>Page 13</b>
<b>Chapter 8 – Tic-Tac-Toe Strategy Tips</b>	<b>Page 14</b>
<b>Chapter 9 – GitHub Account / Repository Creation</b>	<b>Page 15</b>
<b>Chapter 9.1 – GitHub Repository Configuration / Access Copilot</b>	<b>Page 16</b>
<b>Chapter 9.2 – GitHub Copilot / Create File GitHub Repository</b>	<b>Page 17</b>
<b>Chapter 9.3 – Create File GitHub / Add File GitHub Repository</b>	<b>Page 18</b>
<b>Chapter 9.4 – Add File GitHub Repository</b>	<b>Page 19</b>
<b>Chapter 9.5 – Share GitHub Repository</b>	<b>Page 20</b>
<b>Chapter 10 – Summary</b>	<b>Page 21</b>

# 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 payer game.
- (ii) A version with an **AI Opponent**.
- (iii) A project you can upload to GitHub and showcase to the world via **GitHub**.

## Chapter 2

# 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 its 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

### **class TicTacToe**

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

A Python class is similar in structure to a chapter in a book, where the book is the entire program or script, functions are like verses. Each function is contained within a gray block for presentation purposes. Functions are preceded by a commented line instantiated with a hash tag “#” followed by a number identifying its position in **class TicTacToe** and a brief explanation of syntax.

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

#### **class TicTacToe**

And nine functions:

```
__init__()  
display_board()  
is_valid_move()  
make_move()  
check_winner()  
switch_player()  
get_move()  
play()  
main()
```

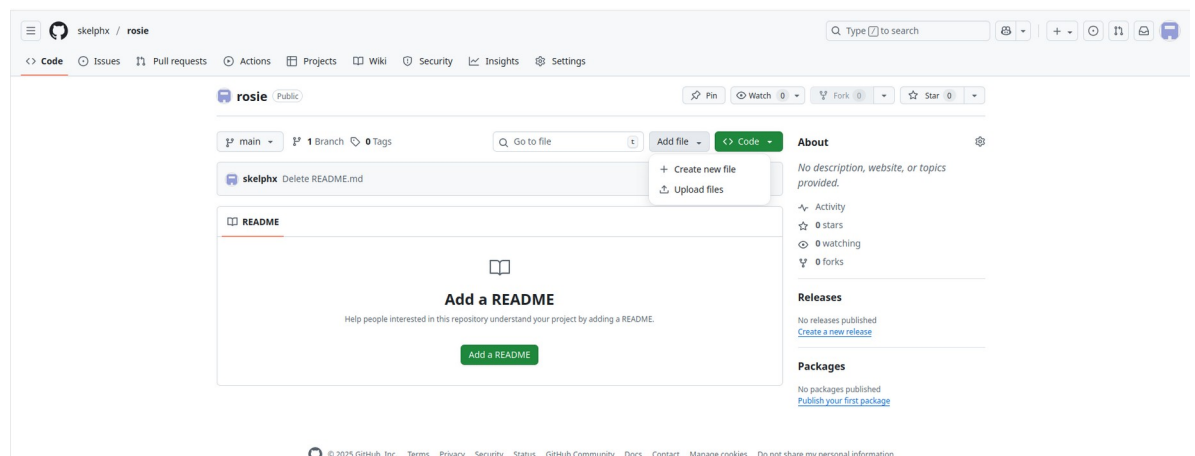


## Chapter 5

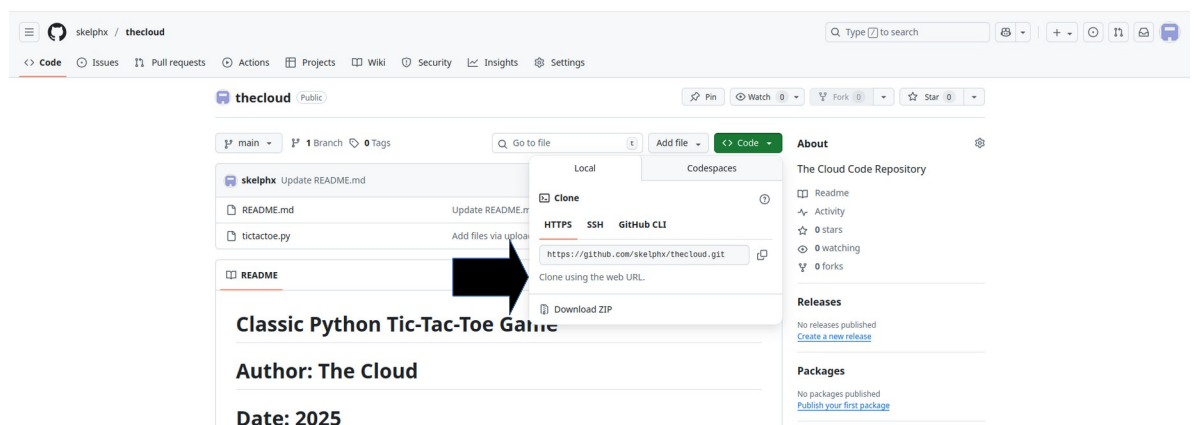
# tictactoe.py

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



- (ii) 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 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

```
1 #3 is_valid_move() if position is empty and between >=0 & <=9 it is a valid position.
2
3 def is_valid_move(self, position):
4     try:
5         pos = int(position) - 1
6         return 0 <= pos <= 8 and self.board[pos] == ' '
7     except ValueError:
8         return False
```

## Block 4

```
1 #4 make_move -1 from positions and play.
2
3 def make_move(self, position):
4     pos = int(position) - 1
5     self.board[pos] = self.current_player
```

## Block 5

```
1 #5 check_winner() after move check winning combination and that all spaces are 2 played, if check proves
2 true, end game, if false do not.
3
4 def check_winner(self):
5     combos = [
6         [0,1,2],[3,4,5],[6,7,8],
7         [0,3,6],[1,4,7],[2,5,8],
8         [0,4,8],[2,4,6]
9     ]
10    for combo in combos:
11        if self.board[combo[0]] == self.board[combo[1]] == self.board[combo[2]] != ' ':
12            self.winner = self.board[combo[0]]
13            self.game_over = True
14            return True
15        if ' ' not in self.board:
16            self.winner = 'Draw'
17            self.game_over = True
18    return True
19    return False
```

## Block 6

```
1 #6 switch_player() switches to other player.
2
3 def switch_player(self):
4     self.current_player = 'O' if self.current_player == 'X' else 'X'
```

# Chapter 5.3

## Block 7

```
1 #7 get_move() allow move to positions 1-9 if 0 positions, else if move is to invalid
2 position display error.
3
4 def get_move(self):
5     while True:
6         move = input(f"Player {self.current_player}, choose (1-9): ")
7         if move.lower() in ['q', 'quit', 'exit']:
8             sys.exit("Thanks for playing!")
9         if self.is_valid_move(move):
10             return move
11         print("Invalid move. Try again.")
```

## Block 8

```
1 #8 play() starts game then only when playing, call display_board(), get_move(),
2 make_move(), check_winner(), switch_player and display_board() again, if
3 check_winner() indicates a winner, exit, print and concatenate Result: with
4 winning player.
5
6 def play(self):
7     input("Press Enter to begin...")
8     while not self.game_over:
9         self.display_board()
10        move = self.get_move()
11        self.make_move(move)
12        if self.check_winner():
13            break
14        self.switch_player()
15        self.display_board()
16        print(f" Result: {self.winner}!")
```

## Block 9

```
1 #9 main() if game is won, decide whether or not to play again.
2
3 def main():
4     while True:
5         game = TicTacToe()
6         game.play()
7         again = input("Play again? (y/n): ").lower()
8         if again != 'y':
9             Break
10        if __name__ == "__main__":
11            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.



```
Tic-Tac-Toe
| |
-----
| |
-----
| |
Player X, choose (1-9):
```

## 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.

```
1 def get_move(self):
2     if self.current_player == 'X':
3         while True:
4             move = input("Your move (1-9): ")
5             if move.lower() in ['q', 'quit', 'exit']:
6                 sys.exit("Game exited.")
7             if self.is_valid_move(move):
8                 return move
9             print("Invalid. Try again.")
10    else:
11        available = [str(i+1) for i, val in enumerate(self.board) if val == ' ']
12        move = random.choice(available)
13        print(f"AI chooses: {move}")
14        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.

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 error messages or warnings and the Tic-Tac-Toe AI game screen is displayed. Can you beat the AI Opponent?

## Chapter 8

# Tic-Tac-Toe Strategy Tips

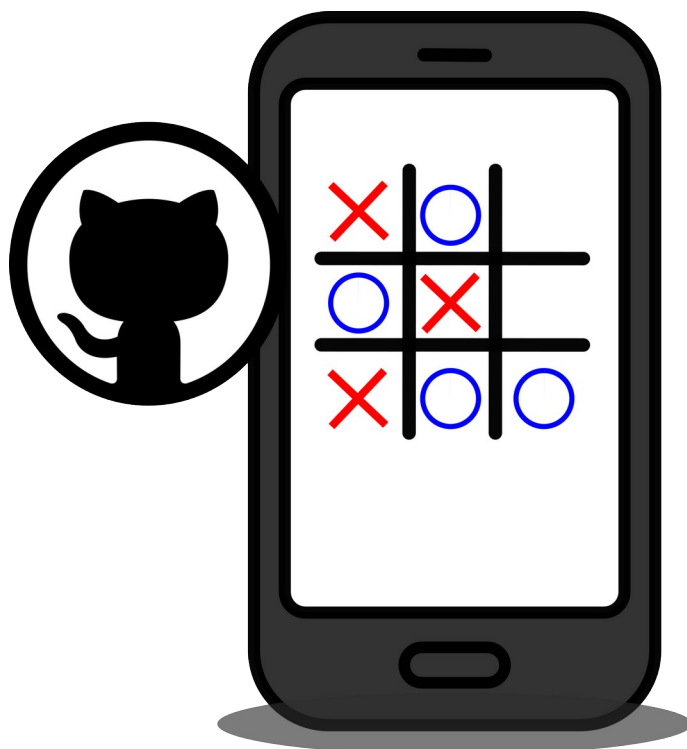
Rule 1: Control the center square.

Rule 2: Block the opponents winning move.

Rule 3: Corners are better than edges,

Rule 4: Avoid giving Forks to an opponent.

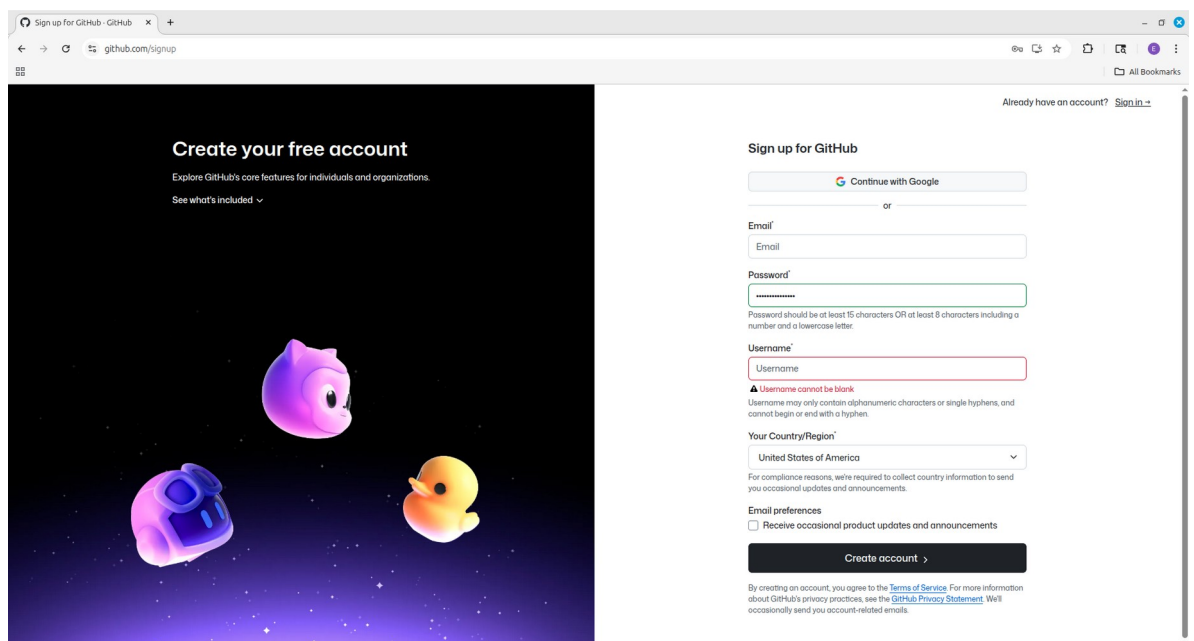
Rule 5: There is no Spoon (Matrix joke lol).



## Chapter 9

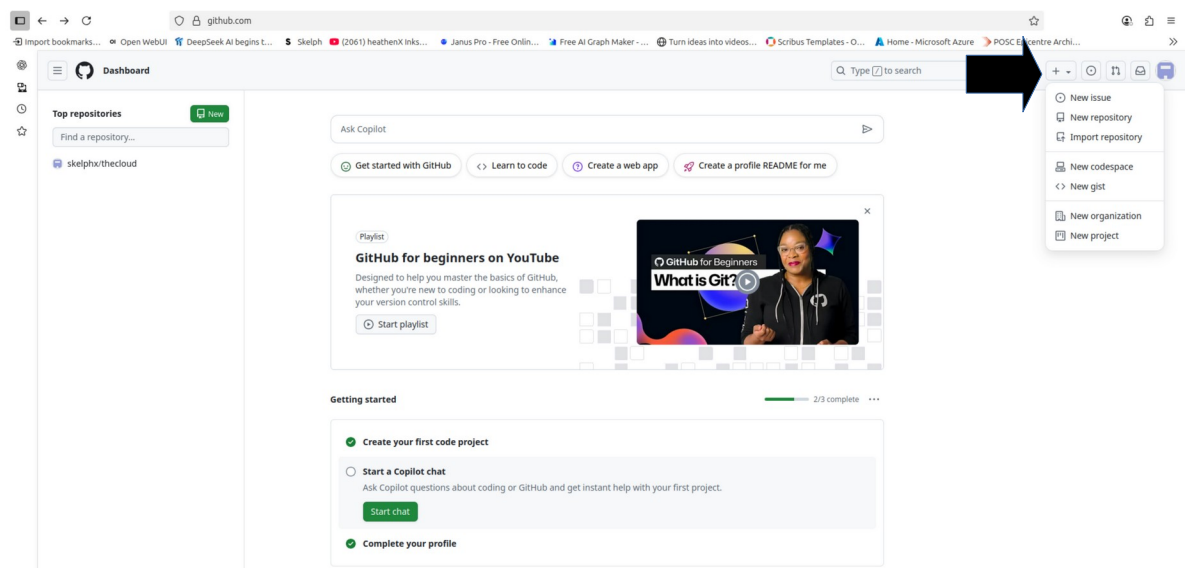
# GitHub Create Account/Repository

(i) Open your browser and type **https://github.com/signup** in the URL window, enter your credentials and click **Create Account**.



The screenshot shows the GitHub sign-up page. On the left, there's a dark blue banner with the text "Create your free account" and "Explore GitHub's core features for individuals and organizations." Below this, there are three GitHub Bots (Octocat, Octocat, and Octocat) floating in space. On the right, the "Sign up for GitHub" form is visible. It includes a "Continue with Google" button, an "Email" field, a "Password" field, and a "Username" field. Below the "Username" field, there's a warning: "Username cannot be blank. Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen." There's also a "Your Country/Region" dropdown menu set to "United States of America". At the bottom, there's an "Email preferences" section with a checkbox for "Receive occasional product updates and announcements". A "Create account" button is at the bottom right.

(ii) Click "+" on GitHub main menu, select **New Repository** from the dropdown.

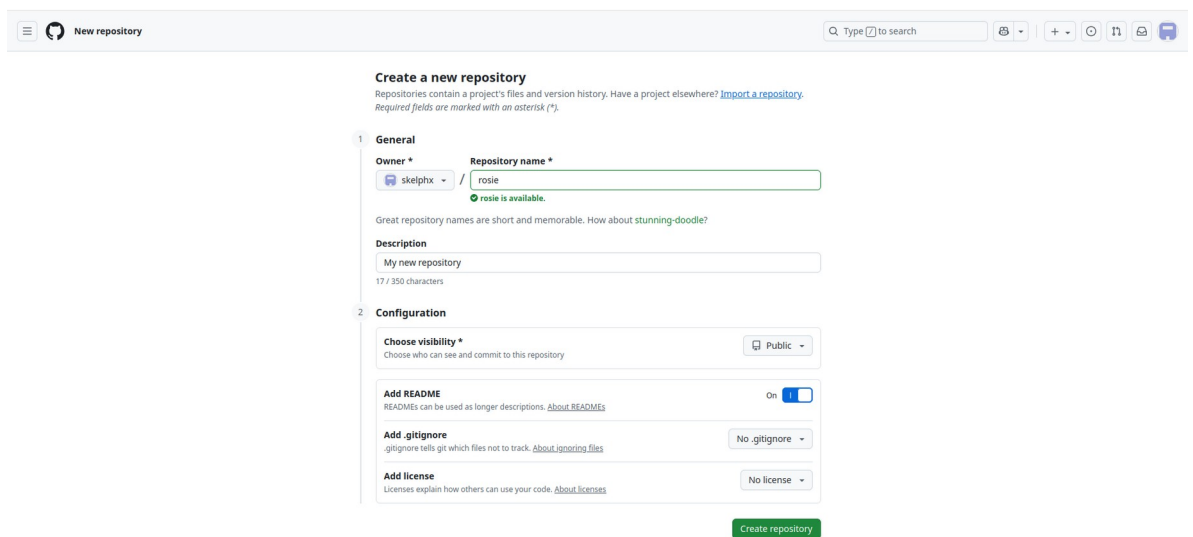




## Chapter 9.1

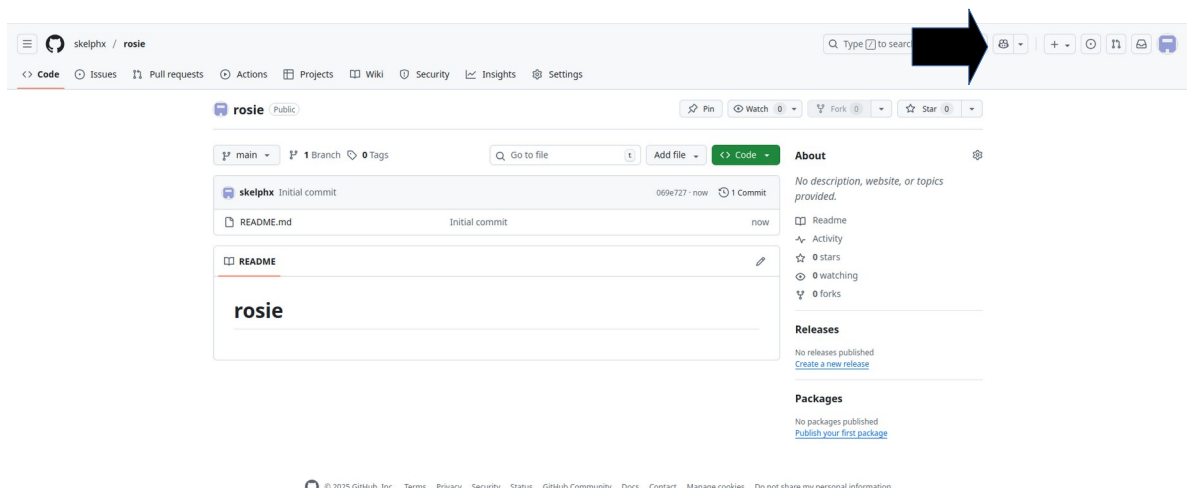
# GitHub Repository Configuration

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



The screenshot shows the 'Create a new repository' page on GitHub. The form is divided into two sections: 'General' and 'Configuration'. In the 'General' section, the 'Owner' is 'skelphx' and the 'Repository name' is 'rosie', which is marked as available. The 'Description' field contains 'My new repository'. In the 'Configuration' section, 'Choose visibility' is set to 'Public'. 'Add README' is turned 'On'. 'Add .gitignore' is set to 'No .gitignore'. 'Add license' is set to 'No license'. A green 'Create repository' button is at the bottom right.

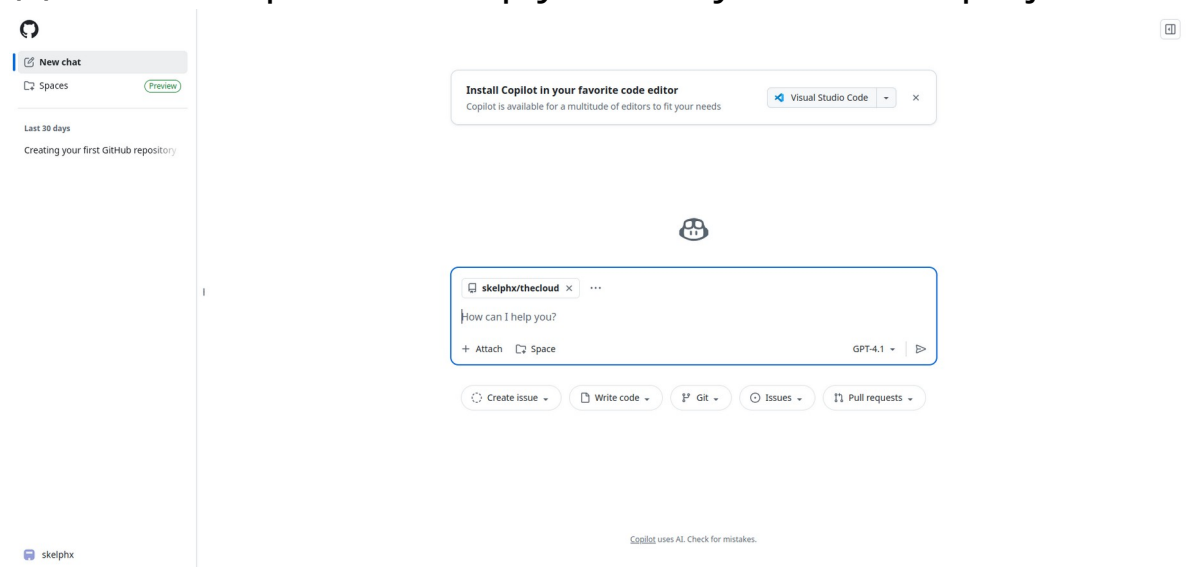
(iv) **Great Job!** – You have successfully created a new GitHub account with public repository where you can share your projects over the Internet. You can access **GitHub Copilot** by clicking the **GitHub Copilot icon** (see below).



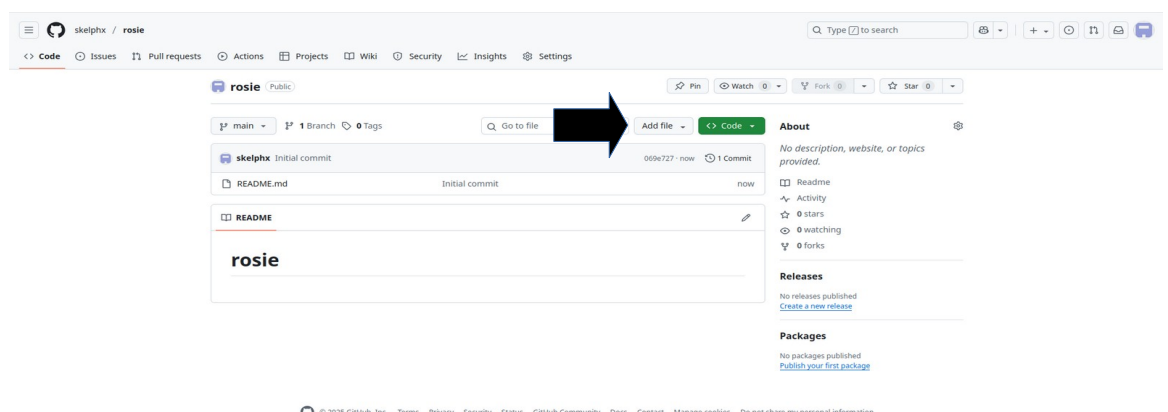
## Chapter 9.2

# CoPilot/Add File to GitHub Repo

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



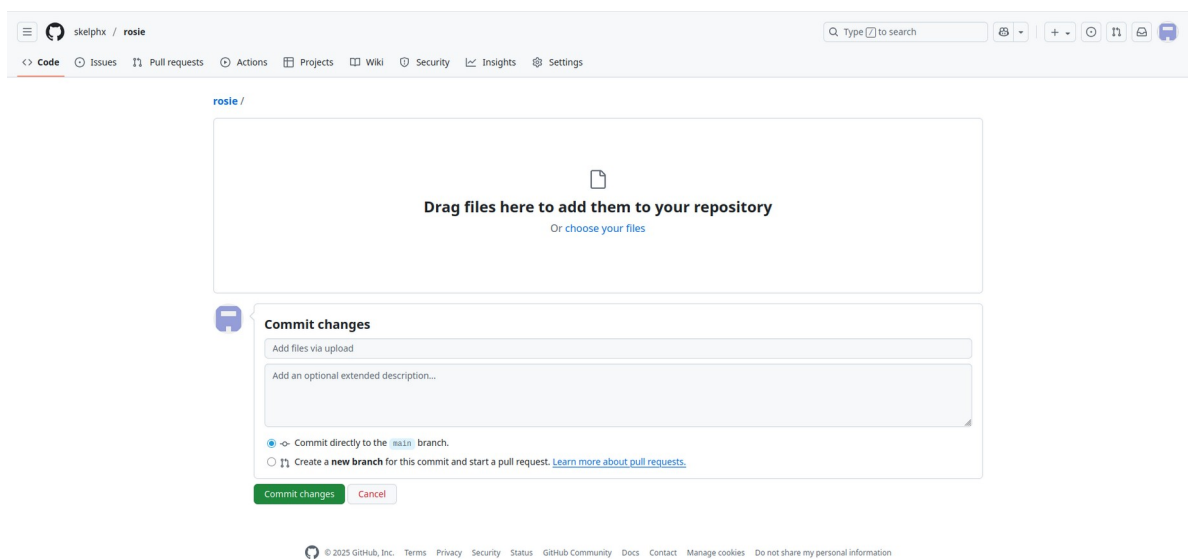
(vi) Click **Add File** from you Repo (see screenshot below) to upload the **tictactoe.py** file you downloaded in **Chapter 5**. You should see the **Add File** dialogue window in **Chapter 9.3**. Files are uploaded to your GitHub Repository from here. This step is unnecessary if you have already created **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

# Add File to GitHub Repo Window

(iii) This is where and how you upload files to your GitHub Repository.



## Chapter 9.4

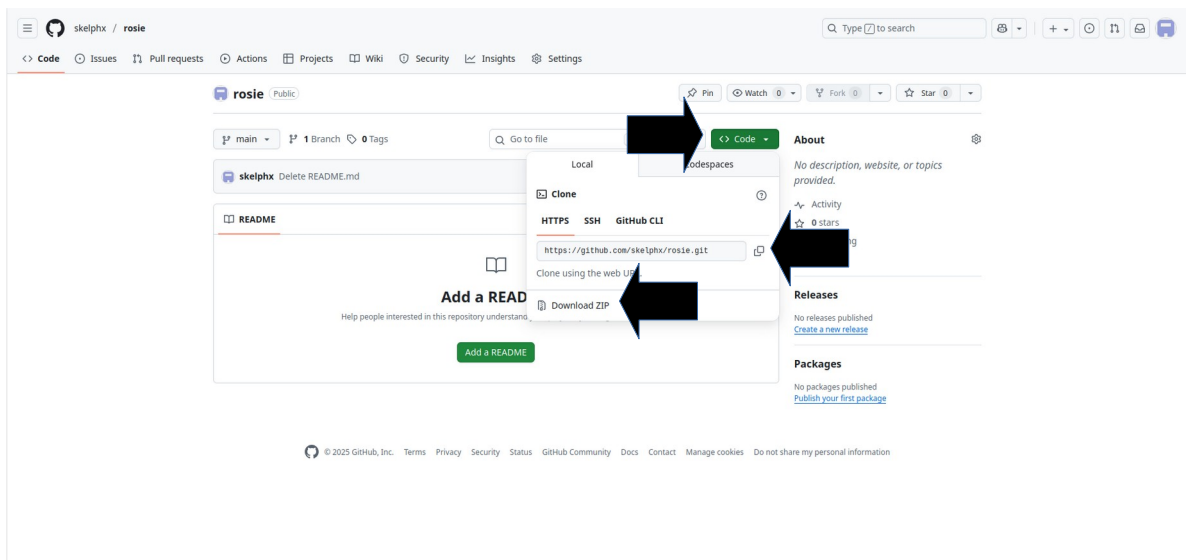
# Congratulations: Your Tic-Toc-Toe game can be shared with the world!

To share your Tic-Tac-Toe game or any other project via GitHub, click the **green code button** on your GitHub Repository (see screenshot) then the **copy button** to the right of the url (`https://github.com/skelphx/rosie.git`) in the drop down window.

Your repository will have a slightly different url than the one displayed in the example above.

Paste your url in emails, documents, text messages etc to share the link to your GitHub Repository.

The ZIP archive containing the project files in your Repository can be downloaded by clicking Download ZIP. See screenshot below.



## Chapter 10

### Summary

If you have completed the tutorial – Well Done!

You have covered ALOT.

- (i) Basic Python syntax and terminology.
- (ii) How to 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) Share a GitHub Repository.
- (vii) Access GitHub Copilot (AI Assistant).

