



# Experience AI with GitHub Copilot.

 GitHub Copilot

Ask a question or type '/' for topics



## GitHub Copilot

⇒ Approx 55% Faster Coding

⇒ 46% code written

⇒ Available for all Businesses

## GitHub Copilot

↳ GitHub Copilot is an AI-powered code completion tool developed by GitHub in collaboration with Open-AI

→ Provide intelligent code suggestion and completions directly within their integrated development environment (IDE)

→ Explaining how AI models are trained to understand programming patterns and generate code snippets

## Working

① Set the GitHub Copilot in your development environment.

② Configure settings and preferences to customize the behavior of GitHub Copilot.

For example

ASR Copilot :- write a Python code to add two numbers

Result

```
def add_numbers(a, b);  
    return a + b
```

# Example usage.

```
if __name__ == "__main__":
```

```
    num1 = 5
```

```
    num2 = 3
```

```
    result = add_numbers(num1, num2)
```

```
    print(f"The sum of {num1} and  
{num2} is {result}")
```



# AI pair programming is here.

75% more fulfilled

**55%**  
faster coding

```
go runtime.go JS days between days
9 // Get average runtime
10 func averageRuntimeInSeconds() {
11     var totalTime int64 = 0
12     var failedRuns int64 = 0
13     for _, run := range runs {
14         if run.Failed {
15             failedRuns++
16         } else {
```

Keep flying with your favorite editor



**46%**  
code written

Now available for  
all businesses

Udemy

# introduction to GitHub Copilot

GitHub Copilot is an AI-powered code completion tool developed by GitHub in collaboration with OpenAI. It is designed to assist developers in writing code by providing intelligent code suggestions and completions directly within their integrated development environment (IDE). GitHub Copilot is trained on a vast amount of publicly available code and uses machine learning models to generate context-aware code suggestions in real-time.

OpenAI Codex Model



GitHub



Public code and text  
on the internet

Provide Editor context

Provide Suggestions

Improve Suggestions

Private Code

A screenshot of a code editor window titled "fetch\_pic.js". The code is as follows:

```
JS fetch_pic.js
1 const fetchNASAPictureOfT
2 return fetch('https://api.nasa.gov/mars/rovers/curiosity/images/latest?api_key=DEMO_KEY')
3   .method: 'GET',
4   headers: {
5     'Content-Type': 'application/json'
6   }
7   .then(response => response.json())
8   .then(json => {
9     return json;
10   })
11
12 }
```

The code editor has a dark theme with syntax highlighting. A status bar at the bottom right shows "Copilot".



# Advanced GitHub Copilot Techniques

We will delve into advanced techniques for harnessing the power of GitHub Copilot. You will learn how to utilize GitHub Copilot for complex code generation tasks and explore its integration with popular programming languages and frameworks. We will also cover advanced features such as context-aware completions and code refactoring suggestions.

1. Harnessing the power of GitHub Copilot for complex code generation tasks:

- Understanding how GitHub Copilot can assist in generating complex code structures and algorithms.
- Exploring techniques to effectively utilize GitHub Copilot for tasks that involve intricate logic and dependencies.
- Leveraging GitHub Copilot to streamline the development of large-scale projects.



## KEY FEATURES OF GitHub CoPilot

1 Context-aware  
code  
suggestions

2 Support for  
multiple languages  
and frameworks

3 Adapts to your  
coding style

4 Continuously  
learns from  
your feedback

Ask Copilot



(?) What is a hash table in JS?

(<>) Email validation regex in JS

## Home

Filter



Learn with a tutorial project

...

### Introduction to GitHub

Get started using GitHub in less than an hour.

### GitHub Pages

Create a site or blog from your GitHub repositories with GitHub Pages.

### Code with Copilot

Develop with AI-powered code suggestions using GitHub Copilot, Codespaces, and VS Code.

### Hello GitHub Actions

Create a GitHub Action and use it in a workflow.

See more tutorial projects

The screenshot shows the Microsoft Copilot interface. At the top, there's a search bar with the placeholder "Type ⌘ to search" and several small icons for file operations. Below the search bar is a "GPT 4o" button. The main area contains two code snippets. The first snippet is a terminal window titled "sh" showing the command "pip install pygame". The second snippet is a code editor window titled "Python" containing the code for a Snake game using the Pygame library. The code includes imports for pygame, time, and random, defines colors, and sets display settings. At the bottom, there's a footer with the text "Ask Copilot" and a circular arrow icon.

```
sh
pip install pygame

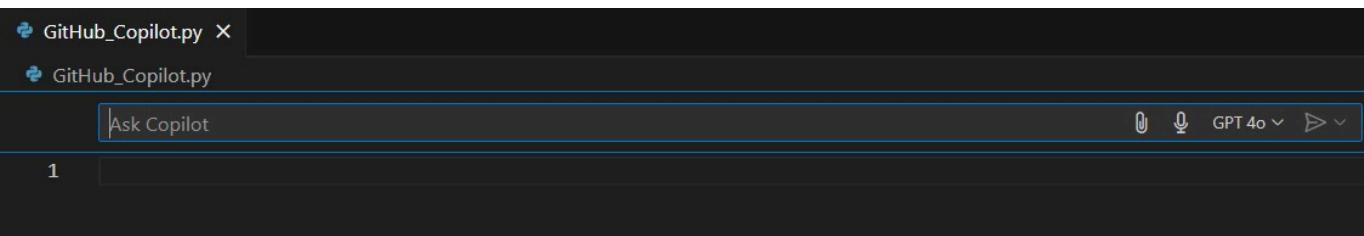
Python

import pygame
import time
import random

pygame.init()

# Define colors
white = (255, 255, 255)
yellow = (255, 255, 102)
black = (0, 0, 0)
red = (213, 50, 86)
green = (0, 255, 0)
blue = (50, 153, 213)

# Display settings
dis_width = 600
dis_height = 400
```



Generate a bar plot between the age and number of people died by heart attack from 2015 to 2018

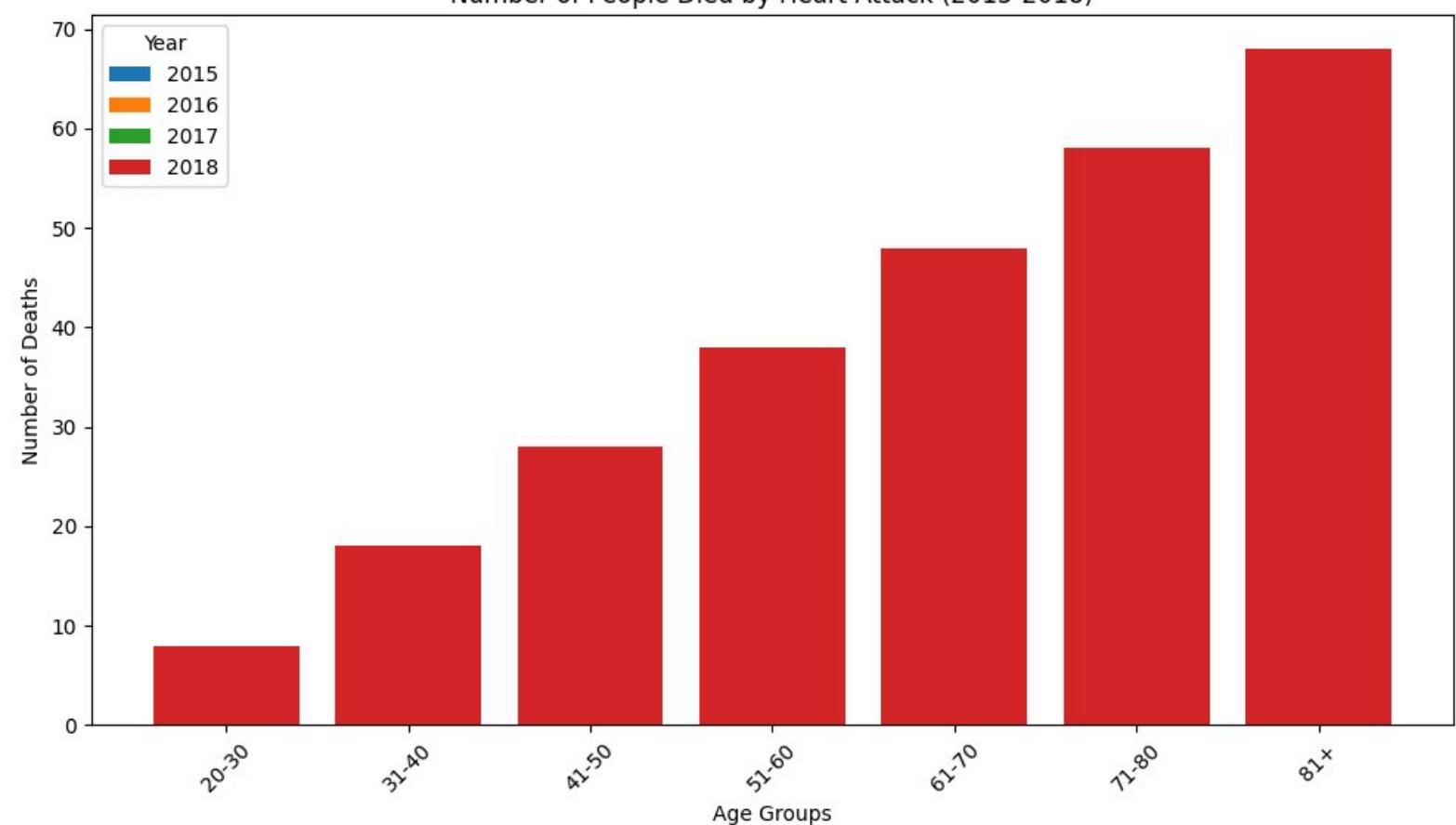
Ask Copilot

GPT 4o ▶

Accept Discard ⌂ ▾

```
1 import matplotlib.pyplot as plt
2
3 # Sample data
4 years = ['2015', '2016', '2017', '2018']
5 age_groups = ['20-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81+']
6 deaths = {
7     '2015': [5, 15, 25, 35, 45, 55, 65],
8     '2016': [6, 16, 26, 36, 46, 56, 66],
9     '2017': [7, 17, 27, 37, 47, 57, 67],
10    '2018': [8, 18, 28, 38, 48, 58, 68]
11 }
12
13 # Plotting
14 fig, ax = plt.subplots(figsize=(10, 6))
15
16 for year in years:
17     ax.bar(age_groups, deaths[year], label=year)
18
19 ax.set_xlabel('Age Groups')
20 ax.set_ylabel('Number of Deaths')
21 ax.set_title('Number of People Died by Heart Attack (2015-2018)')
22 ax.legend(title='Year')
23
24 plt.xticks(rotation=45)
25 plt.tight_layout()
26 plt.show()
```

Number of People Died by Heart Attack (2015-2018)



GitHub\_Copilot.py      Prime\_no\_by\_GC.py 1

Prime\_no\_by\_GC.py > ...

```
1 def prime_number():
2     ## Function to check if a number is prime or not
```

write code to check the given number is prime or not

/fix Expected indented block

3

Prime\_no\_by\_GC.py > prime\_number

```
1 def prime_number():
2     ## Function to check if a number is prime or not
```

write code to check the given number is prime or not

Ask Copilot

0 0 GPT 4o ▶

Accept Discard ⌂ ▾

```
3 num = int(input("Enter a number: "))
4 if num > 1:
5     for i in range(2, int(num**0.5) + 1):
6         if (num % i) == 0:
7             print(num, "is not a prime number")
8             break
9     else:
10        print(num, "is a prime number")
11 else:
12     print(num, "is not a prime number")
```

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for "GitHub\_Copilot.py" and "Prime\_no\_by\_GC.py X". The main area contains the following Python code:

```
1 def prime_number():
2     ## Function to check if a number is prime or not
3     num = int(input("Enter a number: "))
4     if num > 1:
5         for i in range(2, int(num**0.5) + 1):
6             if (num % i) == 0:
7                 print(num, "is not a prime number")
8                 break
9             else:
10                print(num, "is a prime number")
11        else:
12            print(num, "is not a prime number")
13
14 prime_number()
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and COMMENTS. The TERMINAL tab is currently selected. The terminal window displays the following command and output:

```
PS D:\Python codes> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "d:/Python codes/Prime_no_by_GC.py"
Enter a number: 234
234 is not a prime number
PS D:\Python codes>
```



**Snake\_game\_By\_Github\_copilot.py**

```
1 # Generate a python code for a snake game using GitHub Copilot
2 # The code is generated by GitHub Copilot
3 # The GitHub Copilot is an AI pair programmer that helps you write code faster
4
5 import pygame
6 import time
7 import random
8
9 pygame.init()
10
11 # Colors
12 white = (255, 255, 255)
13 red = (255, 0, 0)
14 black = (0, 0, 0)
15
16 # Display settings
17 dis_width = 800
18 dis_height = 600
19 dis = pygame.display.set_mode((dis_width, dis_height))
20 pygame.display.set_caption('Snake Game by GitHub Copilot')
21
22 clock = pygame.time.Clock()
23 snake_block = 10
24 snake_speed = 15
25
26 font_style = pygame.font.SysFont(None, 50)
27 score_font = pygame.font.SysFont(None, 35)
28
29 def our_snake(snake_block, snake_list):
30     for x in snake_list:
31         pygame.draw.rect(dis, red, [x[0], x[1], snake_block, snake_block])
32
33 def message(msg, color):
34     mesg = font_style.render(msg, True, color)
35     dis.blit(mesg, [dis_width / 6, dis_height / 3])
36
37 def gameLoop():
38     game_over = False
39     game_close = False
40
41     x1 = dis_width / 2
42     y1 = dis_height / 2
43
44     x1_change = 0
45     y1_change = 0
46
47     snake_List = []
48     Length_of_snake = 1
```

```
49
50     foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
51     foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
52
53     while not game_over:
54
55         while game_close == True:
56             dis.fill(black)
57             message("You Lost! Press Q-Quit or C-Play Again", red)
58             pygame.display.update()
59
60             for event in pygame.event.get():
61                 if event.type == pygame.KEYDOWN:
62                     if event.key == pygame.K_q:
63                         game_over = True
64                         game_close = False
65                     if event.key == pygame.K_c:
66                         gameLoop()
67
68             for event in pygame.event.get():
69                 if event.type == pygame.QUIT:
70                     game_over = True
71                 if event.type == pygame.KEYDOWN:
72                     if event.key == pygame.K_LEFT:
73                         x1_change = -snake_block
74                         y1_change = 0
75                     elif event.key == pygame.K_RIGHT:
76                         x1_change = snake_block
77                         y1_change = 0
78                     elif event.key == pygame.K_UP:
79                         y1_change = -snake_block
80                         x1_change = 0
81                     elif event.key == pygame.K_DOWN:
82                         y1_change = snake_block
83                         x1_change = 0
84
85                     if x1 >= dis_width or x1 < 0 or y1 >= dis_height or y1 < 0:
86                         game_close = True
87                     x1 += x1_change
88                     y1 += y1_change
89                     dis.fill(black)
90                     pygame.draw.rect(dis, white, [foodx, foody, snake_block, snake_block])
91                     snake_Head = []
92                     snake_Head.append(x1)
93                     snake_Head.append(y1)
94                     snake_List.append(snake_Head)
95                     if len(snake_List) > Length_of_snake:
96                         del snake_List[0]
97
98                     for x in snake_List[:-1]:
```

2/3/25, 11:25 PM

Snake\_game\_By\_Github\_copilot.py

```
99         if x == snake_Head:
100             game_close = True
101
102             our_snake(snake_block, snake_List)
103             pygame.display.update()
104
105             if x1 == foodx and y1 == foody:
106                 foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
107                 foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
108                 Length_of_snake += 1
109
110             clock.tick(snake_speed)
111
112             pygame.quit()
113             quit()
114
115     gameLoop()
```