

# What is Github Copilot?

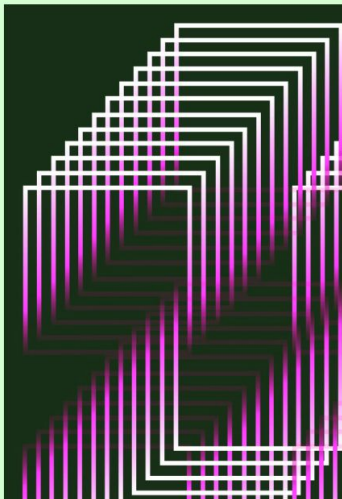
**Discover the code generation model Copilot, to streamline your coding projects.**

# OpenAI Codex suggests code in real-time, right from your editor.

— — —

## Powering next generation applications with OpenAI Codex

Codex is now powering 70 different applications across a variety of use cases through the OpenAI API.

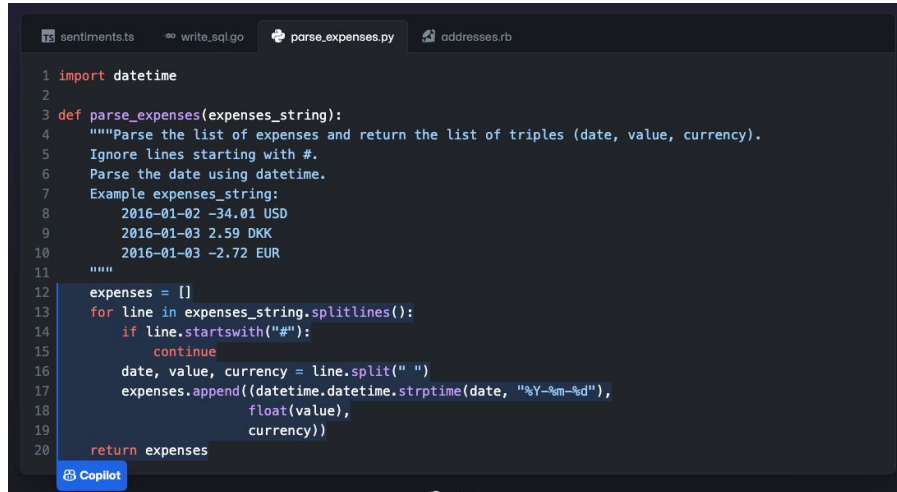


[Source: OpenAI](#)

# Github Copilot

— — —

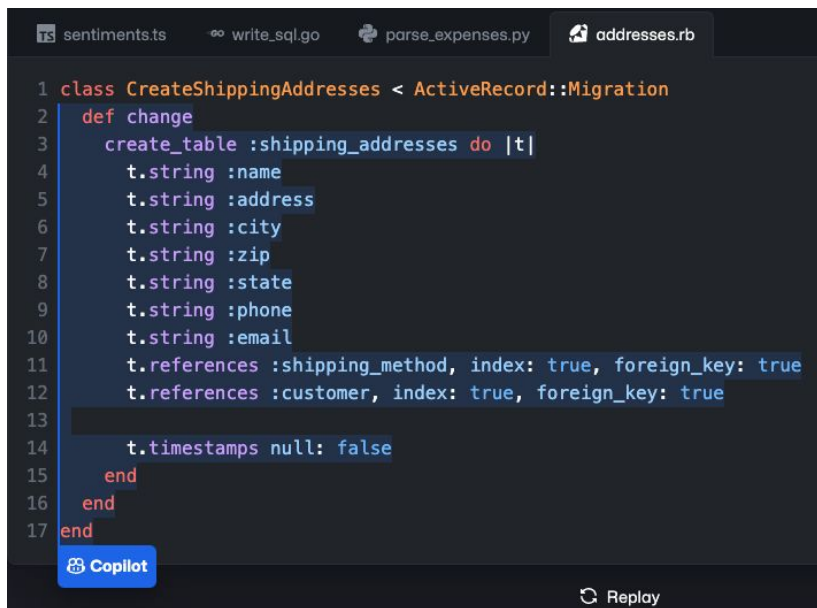
- Created by OpenAI
- Accessed within a coding editor called [Visual Studio Code](#)



```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses
```

# Github Copilot Features

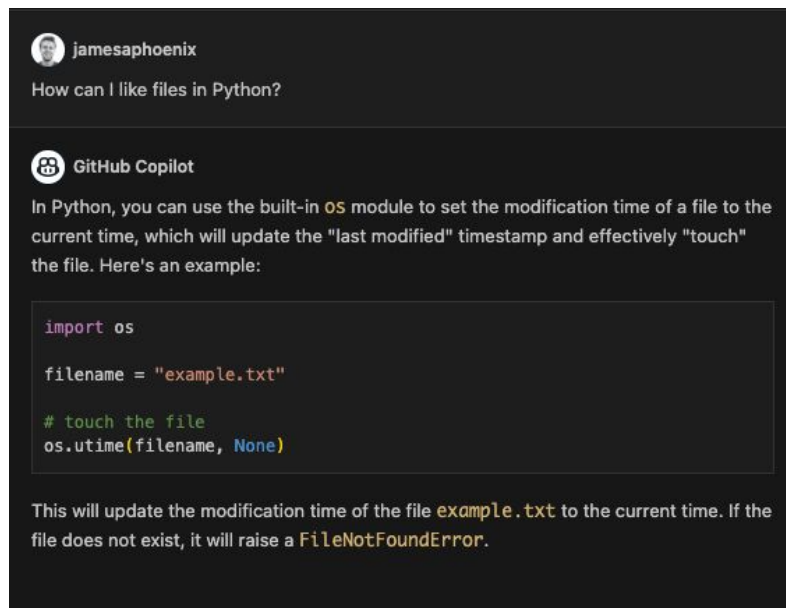
## Generate Code



The screenshot shows a code editor with a dark theme. At the top, there are tabs for 'sentiments.ts', 'write\_sql.go', 'parse\_expenses.py', and 'addresses.rb'. The active file is 'addresses.rb'. The code is a Ruby migration class 'CreateShippingAddresses' that inherits from 'ActiveRecord::Migration'. It has a 'change' method containing a 'create\_table :shipping\_addresses' block. The block defines columns for 'name', 'address', 'city', 'zip', 'state', 'phone', and 'email', all as strings. It also defines foreign keys to 'shipping\_method' and 'customer' tables. The 'timestamps' attribute is set to 'null: false'. A blue Copilot suggestion box is visible at the bottom left of the code editor, and a 'Replay' button is at the bottom right.

```
1 class CreateShippingAddresses < ActiveRecord::Migration
2   def change
3     create_table :shipping_addresses do |t|
4       t.string :name
5       t.string :address
6       t.string :city
7       t.string :zip
8       t.string :state
9       t.string :phone
10      t.string :email
11      t.references :shipping_method, index: true, foreign_key: true
12      t.references :customer, index: true, foreign_key: true
13
14      t.timestamps null: false
15    end
16  end
17 end
```

## Chat



The screenshot shows a chat interface with a dark theme. At the top, there is a user profile for 'jamesaphoenix'. The user's message is 'How can I like files in Python?'. Below the message, there is a response from 'GitHub Copilot'. The response explains that in Python, the 'os' module can be used to set the modification time of a file to the current time, effectively 'touching' the file. It provides an example code snippet in Python. Below the code, it explains that this will update the modification time of the file 'example.txt' to the current time, and that it will raise a 'FileNotFoundError' if the file does not exist.

jamesaphoenix  
How can I like files in Python?

GitHub Copilot

In Python, you can use the built-in `os` module to set the modification time of a file to the current time, which will update the "last modified" timestamp and effectively "touch" the file. Here's an example:

```
import os

filename = "example.txt"

# touch the file
os.utime(filename, None)
```

This will update the modification time of the file `example.txt` to the current time. If the file does not exist, it will raise a `FileNotFoundError`.

# Copilot Use Cases

— — —

- Speeding up your coding projects.
- Debugging code.
- Learning new programming languages and frameworks.

# Why use Github Co-pilot?

— — —

This [research survey](#) showed significant performance boosts to developers using Github co-pilot.