



8

AI tools for code generation

Now generate codes that are just as good, or even better, than human-written code



Swipe >>>

1



GitHub Copilot

- AI code completion tool that helps to write code faster and with fewer errors
- Trained on a massive dataset of code
- Generate code in a variety of programming languages, including Python, Java, JavaScript, and C++

The screenshot shows a code editor with two panels. The left panel displays the GitHub Copilot interface with a prompt: "Write a set of unit test functions for the selected code". Below the prompt, the Copilot has generated Python code for unit testing. The right panel shows the original code being tested, which is a Python function to parse a string of expenses into a list of tuples.

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'), (datetime.datetime(2023, 1, 3), 2.59, 'DKK'), (datetime.datetime(2023, 1, 3), -2.72, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_comments(self):
        expenses_string = '# This is a comment\n2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'), (datetime.datetime(2023, 1, 3), 2.59, 'DKK'), (datetime.datetime(2023, 1, 3), -2.72, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()
```

```
import datetime

def parse_expenses (expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
    2023-01-02 -34.01 USD
    2023-01-03 2.59 DKK
    2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime (date, "%Y-%m-%d"),
                        float (value),
                        currency))

    return expenses

expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
```

Swipe >>>

2



ChatGPT

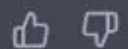
- Exhibits its capability by generating efficient and readable lines of code.
- Optimize the programming process by leveraging pattern analysis in existing code.



I got this when running 'node random-quote-api.js' Error: Cannot find module 'express'



The error message "Cannot find module 'express'" indicates that Node.js is unable to find the required module 'express', which is used to create the web server and handle HTTP requests in the Random Quote Generator API.



To fix this error, you need to install the 'express' module using npm (Node Package Manager). Open your terminal and navigate to the directory where your `random-quote-api.js` file is saved, then run the following command:

Copy code

```
npm install express
```

Swipe >>>



OpenAI Codex

- Used to generate entire programs from natural language descriptions.
- Trained on a massive dataset of code and can generate code in a variety of programming languages, including Python, Java, JavaScript, and Go.

```
parse_expenses.py  write_sql.go  sentiment.ts  addresses.rb

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
```

Swipe >>>



Tabnine

- Offers predictive text and code completion
- Suggest the next lines or blocks of code to speed up the development process
- Grasp the context in which the code is written, providing relevant suggestions even in complex development scenarios

```
1 import os
2 import sys
3
4 # Count lines of code in the given directory, separated by file extension
5 def main(directory):
6     line_count = {}
7     for filename in os.listdir(directory):
8         _, ext = os.path.splitext(filename)
9         if ext not in line_count:
10            line_count[ext] = 0
11            for line in open(os.path.join(directory, filename)):
12                line_count[ext] += 1
13
14 for |
15 ext in line_count.keys():      Tab 12%
16 ext in line_count.keys      Tab+Tab 12%
17 filename in os.listdir      Tab+3 2%
18 ext in                       Tab+4 19%
19 filename                     Tab+5 5%
```

Swipe >>>



Seek

- Used to generate code snippets, functions, and even entire programs from natural language descriptions.
- Trained on a massive dataset of code and can generate code in a variety of programming languages, including Python, Java, JavaScript, and C++.

The screenshot displays the Seek interface with two main panels: 'Questions' and 'Code'. The 'Questions' panel lists several natural language queries, each with a three-dot menu icon to its right. The 'Code' panel shows a SQL query generated in response to the first question. Below the code, a 'Data' section displays a table with two columns: 'Product' and 'Sales'.

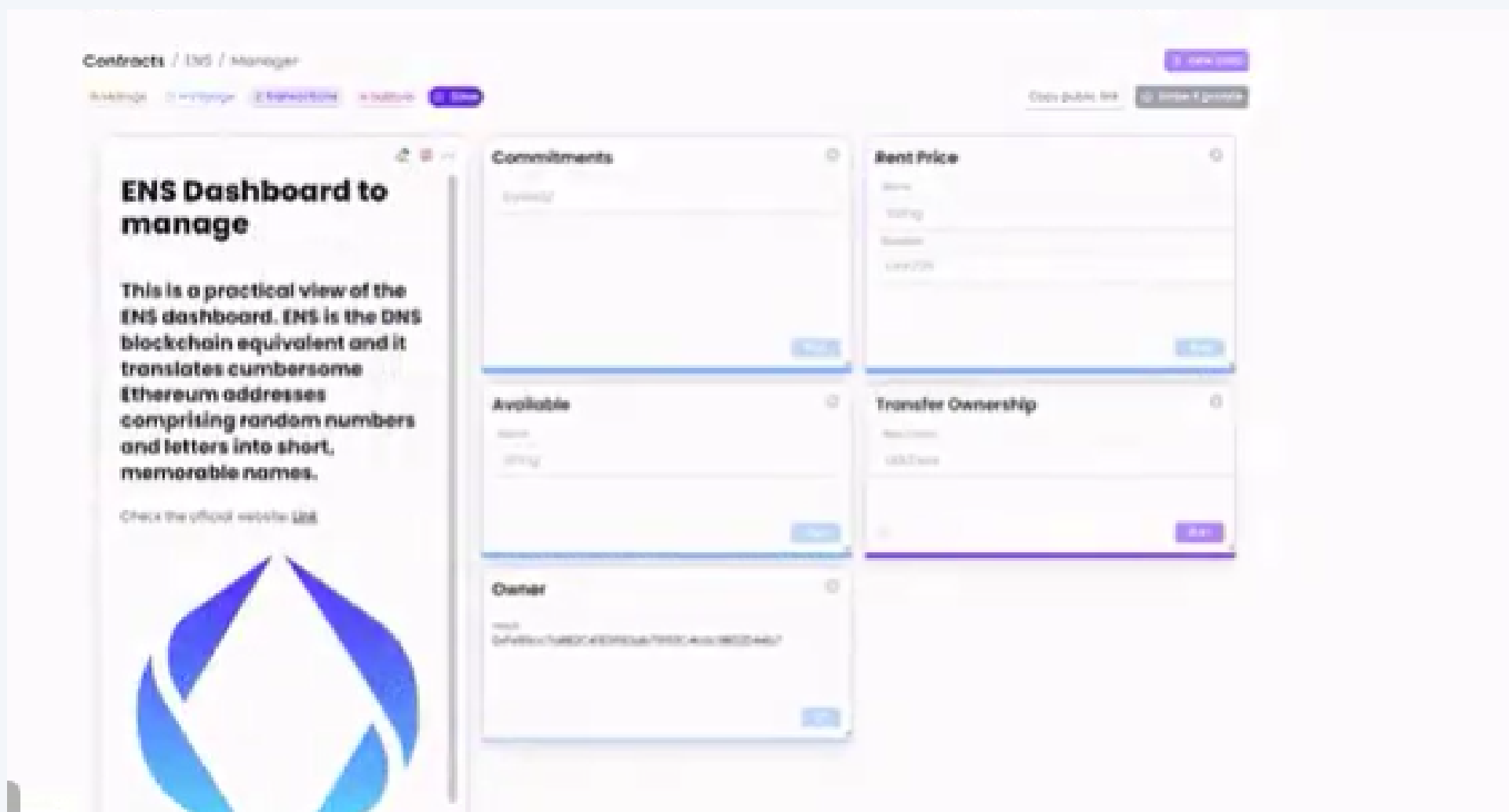
Product	Sales
Product 1	\$155,000
Product 2	\$113,400
Product 3	\$102,050

Swipe >>>



Enzyme

- Designed for frontend web development.
- Used to generate React components, HTML, and CSS from natural language descriptions.
- Build better tools to bring web3 closer to creators



Swipe >>>



Kite

- Aims to assist software developers by providing them with relevant code suggestions in real-time
- Uses Machine Learning models to predict the next line of code
- Popular for its Co-Pilot feature, which assists developers in suggesting entire lines or blocks of code

```
import os
import sys

def count_py_files_in_repos(dirname):
    if os.path.exists(os.path.join(dirname, '.git')):
        count = 0
        for root, dirs, files in os.walk(dirname):
            count += len([f for f in files if f.endswith('.py')])
        print('{} has {} Python files'.format(dirname, count))
    for name in os.listdir(dirname):
        path = os.path.join(dirname, name)
        if os.path.isdir(path):
            cou|
                count_py_files_in_repos(...)
            count
```

Swipe >>>



Codota

- Provides intelligent code completion, suggestions, and inline documentation by learning from existing code.
- Analyze code from various repositories and learning patterns.
- Supports various languages and can be integrated into IDEs like IntelliJ IDEA, Android Studio, and Eclipse

```
public static void main(String[] args) throws Exception {
    String filePath = args[0];
    File file = n|
}

>. new File(filePath)
  new
↑ null
↓ m new File(String pathname)
↑ >. new File(filePath, String child)
↓ m new File(String parent, String child)
↓ m new File(File parent, String child)
  m new File(URI uri)
>. new File(System.getProperty("java.io.tmpdir"))
>. new File(String parent, "lib")
```

Swipe >>>

Want to Build a Customized LLM Application?

Check out our in-person large language models bootcamp!



Seattle

January 29 - February 2,
2024



[Learn More](#)



Online

Coming Soon