

Generating Code



This section is under development.

LLMs like ChatGPT are very effective at code generation. In this section, we will cover many examples of how to use ChatGPT for code generation.

The OpenAI's Playground (Chat Mode) and the `gpt-3.5-turbo` model are used for all examples below.

As with all chat models from OpenAI, you can use a `System Message` to define the behavior and format of the responses. We will use the following System Message for the prompt examples:

```
You are a helpful code assistant that can teach a junior developer how to code. Your language of choice is Python. Don't explain the code, just generate the code block itself.
```

Basic Example

The `User Message` will then be the prompt itself with the instruction of the specific code you want the model to generate. Below is a snapshot of the first basic prompt for code generation. Note that the final code was generated by the `Assistant`.

SYSTEM

You are a helpful code assistant that can teach a junior developer how to code. Your language of choice is Python. Don't explain the code, just generate the code block itself.

USER

Write code that asks the user for their name and say "Hello"

ASSISTANT

```
'''  
name = input("What is your name? ")  
print("Hello", name)  
'''
```

⊕ Add message

Mode

Chat Beta ▾

Model

gpt-3.5-turbo ▾

Temperature 0.7

Maximum length 256

Top P 1

Frequency penalty 0

Presence penalty 0

Here is the prompt if you want to try it yourself:

Write code that asks the user for their name and say "Hello"

Turn Comments Into Code

You can also use the code generation capabilities of these LLMs to generate code from comments alone. Let's look at another example that passes the instructions as a comment block:

Prompt:

```
'''  
1. Create a list of movies  
2. Create a list of ratings for these movies  
3. Combine them to make a json object of 10 movies with their ratings.  
'''
```

Output:

```
movies = ["The Shawshank Redemption", "The Godfather", "The Dark Knight", "Schindler's List",  
"Pulp Fiction", "The Lord of the Rings: The Return of the King", "Forrest Gump", "Star Wars:  
Episode V - The Empire Strikes Back", "Inception", "The Silence of the Lambs"]
```

```
ratings = [9.3, 9.2, 9.0, 8.9, 8.9, 8.9, 8.8, 8.7, 8.7, 8.6]
```

```
movie_ratings = {}

for i in range(len(movies)):
    movie_ratings[movies[i]] = ratings[i]

json_object = json.dumps(movie_ratings, indent=4)

print(json_object)
```

The generated code looks like working code. However, the output doesn't contain an important `import` statement, i.e., `import json`. While these models generate working code, you must pay close attention to small details like this and always make sure to test your code.

Complete Functions or Next Line

These LLMs have also been incorporated into tools like GitHub Copilot which makes them useful for developers. One useful feature is the ability of the model to complete functions.

Prompt:

```
# function to multiply two numbers and add 75 to it

def multiply(
```

Output:

```
a, b):
    result = a * b
    result += 75
    return result
```

MySQL Query Generation

Besides the basic code generation example above, you can use the model to generate useful code that could be used in other aspects of programming like creating and testing MySQL queries.

Let's say you have a dataset with some information which you can include as part of the prompt and then instruct it to generate a specific query. For example:

```
"""
Table departments, columns = [DepartmentId, DepartmentName]
Table students, columns = [DepartmentId, StudentId, StudentName]
Create a MySQL query for all students in the Computer Science department
"""
```

Output:

```
SELECT students.StudentId, students.StudentName
FROM students
INNER JOIN departments
ON students.DepartmentId = departments.DepartmentId
WHERE departments.DepartmentName = 'Computer Science';
```

The model generates a query! However, in this example we made up the database so we don't have a database to test the generated query. If you are testing this example, it could be useful to have this database created. Fortunately, you can use ChatGPT to generate this for you as well. One approach is using a modified version of the prompt above with slightly different instructions to generate a database schema:

```
Table departments, columns = [DepartmentId, DepartmentName]
Table students, columns = [DepartmentId, StudentId, StudentName]
Create a valid database schema with the above tables and columns
```

The model generated the following schema:

```
CREATE TABLE departments (
  DepartmentId INT PRIMARY KEY,
  DepartmentName VARCHAR(50)
);
CREATE TABLE students (
  DepartmentId INT,
  StudentId INT PRIMARY KEY,
  StudentName VARCHAR(50),
  FOREIGN KEY (DepartmentId) REFERENCES departments(DepartmentId)
);
```

You can create a temporary database using something like sqliteonline.com.

You will also need dummy data to test if the original generated query was valid. Let's use ChatGPT for this too! The prompt could look something like the following:

```
CREATE TABLE departments (  
  DepartmentId INT PRIMARY KEY,  
  DepartmentName VARCHAR(50)  
);  
CREATE TABLE students (  
  DepartmentId INT,  
  StudentId INT PRIMARY KEY,  
  StudentName VARCHAR(50),  
  FOREIGN KEY (DepartmentId) REFERENCES departments(DepartmentId)  
);  
Given the database schema above, generate valid insert statements include 4 rows for each table.
```

The output with the insert statements:

```
INSERT INTO departments (DepartmentId, DepartmentName)  
VALUES (1, 'Computer Science'), (2, 'Mathematics'), (3, 'Physics'), (4, 'Biology');  
INSERT INTO students (DepartmentId, StudentId, StudentName)  
VALUES (1, 101, 'John Doe'), (1, 102, 'Jane Doe'), (2, 201, 'Alice Smith'), (2, 202, 'Bob Johnson');
```

Now that you have the database and the dummy data, you can test the original query. Based on the dummy data, the result should be two rows with `Jane Doe` and `John Doe`.

Explain Code

If you are learning to program in a certain language, it might be useful to prompt the model to explain certain bits of code. Let's reuse the query generated above and ask the model to explain it.

If you are using the same System Message from before, be careful of how you are instructing the model. For example, if we use the following example, the model will complain that it is a code assistant in Python but it still provides an explanation. Sometimes the model refuses to generate what you are instructing because it's designed to follow the System Message.

Prompt:

```
SELECT students.StudentId, students.StudentName
```

```
FROM students
INNER JOIN departments
ON students.DepartmentId = departments.DepartmentId
WHERE departments.DepartmentName = 'Computer Science';
Explain the above SQL statement.
```

Output:

Sorry, as a code assistant in Python, I cannot generate SQL code. However, I can explain the purpose of the SQL statement provided.

This SQL statement selects the `StudentId` and `StudentName` columns from the `students` table, where the `DepartmentName` column in the `departments` table is equal to 'Computer Science'. To achieve this, the statement performs an inner join on the `DepartmentId` column between the `students` and `departments` tables.

In simpler terms, it is selecting the names and IDs of students who are enrolled in the Computer Science department.

Editing Code

Example coming soon!

Debugging Code

Example coming soon!

Best practices

Coming soon!

Last updated on September 19, 2024