GitHub Copilot

Guided Practice

QUESTIONS

What is GitHub Copilot primarily designed for?

- A) Debugging code
- B) Code completion and suggestion
- C) Code versioning
- D) Testing frameworks

Answer: B) Code completion and suggestion

QUESTIONS

Which of the following features allows GitHub Copilot to generate boilerplate code?

- A) Code reviews
- B) Contextual code suggestions
- C) Manual coding
- D) Repository cloning

Answer: B) Contextual code suggestions

QUESTIONS

In which programming languages does GitHub Copilot provide the best support?

- A) Only JavaScript
- B) Java and C++
- C) Multiple languages including Python, JavaScript, and TypeScript
- D) Only Python

Answer: C) Multiple languages including Python, JavaScript, and TypeScript

QUESTIONS

What is a primary benefit of using templates in GitHub Copilot?

- A) Slower development
- B) Enhanced collaboration
- C) Faster onboarding for new developers
- D) Increased manual coding

Answer: C) Faster onboarding for new developers

QUESTIONS

How does GitHub Copilot generate documentation?

- A) By manually writing comments
- B) By analyzing code comments and structures
- C) By copying from other projects
- D) It cannot generate documentation

Answer: B) By analyzing code comments and structures

QUESTIONS

Which of the following is a potential downside of relying on GitHub Copilot for boilerplate code?

- A) Increased productivity
- B) Less flexibility in coding style
- C) Provides context-aware suggestions
- D) Code might not follow best practices

Answer: D) Code might not follow best practices

QUESTIONS

What is the first step to integrate GitHub Copilot into an existing project?

- A) Write the entire code manually
- B) Install the GitHub Copilot extension
- C) Create a new repository
- D) Upgrade the IDE

Answer: B) Install the GitHub Copilot extension

QUESTIONS

What type of boilerplate code can GitHub Copilot help generate?

- A) Database schemas
- B) API endpoints
- C) HTML templates
- D) All of the above

Answer: D) All of the above

QUESTIONS

Which feature of GitHub Copilot allows for customizing code suggestions?

- A) User feedback on suggestions
- B) Manual overrides
- C) Preference settings in the IDE
- D) All of the above

Answer: D) All of the above

QUESTIONS

How can automated documentation generation improve project maintainability?

- A) By reducing code clarity
- B) By ensuring up-to-date references
- C) By increasing complexity
- D) By complicating onboarding

Answer: B) By ensuring up-to-date references

.

QUESTIONS

What would the following GitHub Copilot prompt generate for boilerplate code?

```
// Create an Express server
```

- A) Basic HTTP server code
- B) React component
- C) A database connection
- D) A CSS stylesheet

Answer: A) Basic HTTP server code

QUESTIONS

Which of the following snippets can be generated by Copilot for creating a REST API endpoint in JavaScript?

```
app.get('/api/users', (req, res) => {
    // Fetch users from database
});
```

- A) Creates an empty function
- B) Generates a full API endpoint with comments
- C) Only creates a database connection
- D) Returns a static user list

Answer: B) Generates a full API endpoint with comments

QUESTIONS

If you wanted to generate documentation for a function using Copilot, which comment style would you use?

```
/**
 * Function to add two numbers
 * @param {number} a - first number
 * @param {number} b - second number
 * @returns {number} - sum of a and b
 */
function add(a, b) {
    return a + b;
}
```

- A) Single-line comment
- B) JSDoc comment
- C) Markdown comment
- D) No comment required

Answer: B) JSDoc comment

QUESTIONS

What code does GitHub Copilot suggest when you type `function fetchData`?

- A) An empty function
- B) Code to fetch data from an API
- C) Code to handle errors
- D) A function declaration with no parameters

Answer: B) Code to fetch data from an API

QUESTIONS

When integrating GitHub Copilot, what command is typically used to install the necessary dependencies for an Express app?

```
npm install express
```

- A) npm install express-generator
- B) npm install express
- C) npm add express
- D) npm update express

Answer: B) npm install express

QUESTIONS

What is the primary role of GitHub Copilot in debugging code?

- A) Replace human testers
- B) Provide code suggestions to fix errors
- C) Compile the code
- D) Run performance tests

Answer: B) Provide code suggestions to fix errors

QUESTIONS

How can GitHub Copilot assist in writing test cases?

- A) By generating code snippets based on function signatures
- B) By executing tests automatically
- C) By deleting unnecessary tests
- D) By analyzing test coverage

Answer: A) By generating code snippets based on function signatures

QUESTIONS

Which testing framework is commonly suggested by GitHub Copilot for writing unit tests in JavaScript?

- A) Mocha
- B) React
- C) Express
- D) TypeScript

Answer: A) Mocha

QUESTIONS

Which of the following is a security consideration when using AI-generated code?

- A) Code optimization
- B) Reviewing for security vulnerabilities
- C) Enhancing code readability
- D) Reducing the number of lines of code

Answer: B) Reviewing for security vulnerabilities

QUESTIONS

What test case might GitHub Copilot generate for a function that adds two numbers?

```
function add(a, b) {
    return a + b;
}
```

- A) It returns the sum of two numbers
- B) It returns the difference of two numbers
- C) It logs the parameters
- D) It does not generate any test case

Answer: A) It returns the sum of two numbers

QUESTIONS

Given the following function, what would be a typical Copilot suggestion for a test case?

```
function divide(a, b) {
    return a / b;
}
```

- A) Test with a zero divisor
- B) Test with negative numbers
- C) Both A and B
- D) No tests required

Answer: C) Both A and B

QUESTIONS

Which of the following test frameworks does GitHub Copilot commonly suggest for testing asynchronous functions?

```
async function fetchData(url) {
    const response = await fetch(url);
    return response.json();
}
```

- A) Jest
- B) Jasmine
- C) Mocha
- D) All of the above

Answer: D) All of the above

QUESTIONS

If you encounter a logical error in your code, how can GitHub Copilot assist?

```
function calculateTotal(items) {
    return items.reduce((total, item) => total + item.price, 0);
}
```

- A) It can identify the error and suggest fixes
- B) It can only provide syntax corrections
- C) It will not provide any assistance
- D) It will rewrite the entire function

Answer: A) It can identify the error and suggest fixes

QUESTIONS

What might GitHub Copilot suggest as a way to handle errors in a promise-based function?

```
fetchData(url).then(data => {
    console.log(data);
});
```

- A) Ignore the error
- B) Use a `.catch()` method
- C) Rewrite the function
- D) Return null

Answer: B) Use a `.catch()` method

QUESTIONS

Given the following function, what test case might GitHub Copilot suggest to test for potential errors?

```
function getUser(id) {
    return fetch(`/api/users/${id}`)
        .then(response => response.json());
}
```

- A) Test with a valid user ID
- B) Test with an invalid user ID
- C) Test with a non-numeric ID
- D) All of the above

Answer: D) All of the above

QUESTIONS

What kind of error handling might Copilot suggest for this asynchronous function?

```
async function fetchData(url) {
    const response = await fetch(url);
    return response.json();
}
```

- A) Add a try-catch block to handle potential errors
- B) Ignore errors since they don’t affect functionality
- C) Return a hardcoded value
- D) Rewrite the function in synchronous style

Answer: A) Add a try-catch block to handle potential errors

QUESTIONS

How might GitHub Copilot help in writing a unit test for this function using Jest?

```
function multiply(a, b) {
    return a * b;
}
```

- A) Suggest a test case like `expect(multiply(2, 3)).toBe(6);`
- B) Suggest removing the test
- C) Suggest testing only with strings
- D) Suggest a different function name

Answer: A) Suggest a test case like `expect(multiply(2, 3)).toBe(6);`

QUESTIONS

If the following code snippet has a potential security flaw, what might GitHub Copilot recommend?

```
function submitForm(data) {
    fetch('/api/submit', {
        method: 'POST',
        body: JSON.stringify(data),
        headers: {
            'Content-Type': 'application/json'
        }
    });
}
```

- A) Validate the data before submitting
- B) Submit the form without validation
- C) Use GET method instead
- D) Hardcode data for testing

QUESTIONS

What testing strategy might Copilot suggest for ensuring that error scenarios are covered in your tests?

```
function processPayment(amount) {
    if (amount <= 0) throw new Error("Invalid amount");
    // process payment logic
}
```

- A) Only test with valid amounts
- B) Test with zero and negative amounts to ensure the error is thrown
- C) Skip testing altogether
- D) Use console logs instead of throwing errors

Answer: B) Test with zero and negative amounts to ensure the error is thrown

QUESTIONS

What is the primary function of GitHub Copilot when writing JavaScript functions?

- A) Debugging code
- B) Providing suggestions and autocompletion
- C) Running tests
- D) Refactoring code

Answer: B) Providing suggestions and autocompletion

QUESTIONS

How does GitHub Copilot assist in writing method signatures?

- A) By generating random signatures
- B) By analyzing existing code and context
- C) By providing documentation links
- D) By copying from other projects

Answer: B) By analyzing existing code and context

QUESTIONS

What is comment-driven code generation in GitHub Copilot?

- A) Writing comments without code
- B) Generating code based on comments provided by the developer
- C) Commenting out existing code
- D) Writing comments after code is complete

Answer: B) Generating code based on comments provided by the developer

QUESTIONS

Which of the following is a benefit of using GitHub Copilot for refactoring code?

- A) Decreased code quality
- B) Faster identification of code smells
- C) Eliminates the need for testing
- D) Requires less understanding of the codebase

Answer: B) Faster identification of code smells

QUESTIONS

What type of suggestions can GitHub Copilot provide when writing functions?

- A) Full function implementations
- B) Partial code snippets
- C) Input validation techniques
- D) All of the above

Answer: D) All of the above

QUESTIONS

Which of the following can GitHub Copilot help improve when writing methods?

- A) Method performance
- B) Code readability
- C) Code complexity
- D) Both A and B

Answer: D) Both A and B

QUESTIONS

What type of context does GitHub Copilot consider for autocompletion of method signatures?

- A) Only the last function written
- B) The entire code file
- C) Global variables only
- D) Code comments

Answer: B) The entire code file

QUESTIONS

How can GitHub Copilot assist with error handling in functions?

- A) By ignoring potential errors
- B) By suggesting try-catch blocks and validation logic
- C) By logging errors to the console
- D) By automatically fixing errors

Answer: B) By suggesting try-catch blocks and validation logic

QUESTIONS

Which of the following is a potential downside of using Copilot for writing functions?

- A) Increased productivity
- B) Generating code that does not follow best practices
- C) Enhanced collaboration
- D) Faster debugging

Answer: B) Generating code that does not follow best practices

QUESTIONS

What feature of GitHub Copilot aids in writing documentation alongside code?

- A) Comment-driven code generation
- B) Manual documentation only
- C) Automatic code formatting
- D) Version control integration

Answer: A) Comment-driven code generation

QUESTIONS

What code might GitHub Copilot suggest for a function that calculates the square of a number?

```
function square(n) {
    // return the square of n
}
```

- A) `return n * n;`
- B) `return Math.pow(n, 2);`
- C) `return n ** 2;`
- D) All of the above

Answer: D) All of the above

QUESTIONS

What autocompletion might Copilot provide for this method signature?

```
class Calculator {
    add(a, b) {
        // implementation here
    }
}
```

- A) `return a + b;`
- B) `console.log(a + b);`
- C) `let result = a + b;`
- D) All of the above

Answer: D) All of the above

QUESTIONS

Given the following comment, what code might Copilot generate?

```
// Function to fetch user data from an API
function fetchUserData(userId) {
    // implementation here
}
```

- A) Fetching code with error handling
- B) A static response
- C) A hardcoded user object
- D) Nothing, since it's a comment

Answer: A) Fetching code with error handling

QUESTIONS

What refactoring suggestion might Copilot make for this function?

```
function isEven(num) {
    if (num % 2 === 0) {
        return true;
    } else {
        return false;
    }
}
```

- A) `return num % 2 === 0;`
- B) `return !num % 2;`
- C) Remove the function entirely
- D) Add console logs

Answer: A) `return num % 2 === 0;`

QUESTIONS

What might GitHub Copilot suggest for the following function with missing error handling?

```
function divide(a, b) {
    return a / b;
}
```

- A) `if (b === 0) throw new Error("Division by zero");`
- B) Just return `0` if `b` is `0`
- C) Ignore the error entirely
- D) Rewrite the function as a multiplication

Answer: A) `if (b === 0) throw new Error("Division by zero");`

QUESTIONS

For this method, what might Copilot suggest to improve its readability?

```
function calculateTotal(prices) {
    let total = 0;
    prices.forEach(price => {
        total += price;
    });
    return total;
}
```

- A) Use `reduce` instead of `forEach`
- B) Remove the function and use a loop
- C) Add more parameters
- D) Leave it as is

Answer: A) Use `reduce` instead of `forEach`

QUESTIONS

What might Copilot suggest as a test case for this function?

```
function greet(name) {
    return `Hello, ${name}!`;
}
```

- A) `expect(greet('Alice')).toBe('Hello, Alice!');`
- B) `greet();`
- C) `greet(123);`
- D) None of the above

Answer: A) `expect(greet('Alice')).toBe('Hello, Alice!');`

QUESTIONS

Given this function, what might Copilot suggest to improve its structure?

```
function processArray(arr) {
    arr.forEach(item => {
        if (item > 10) {
            console.log(item);
        }
    });
}
```

- A) Use `filter` to create a new array of items greater than 10
- B) Just console log without any conditions
- C) Use a while loop instead of forEach
- D) Add console logs to indicate when an item is processed

Answer: A) Use `filter` to create a new array of items greater than 10

QUESTIONS

What code completion might Copilot provide for this function definition?

```
function multiply(a, b) {
    // multiply a and b
}
```

- A) `return a * b;`
- B) `return a + b;`
- C) `return a - b;`
- D) None of the above

Answer: A) `return a * b;`

QUESTIONS

For this async function, what error handling might Copilot suggest?

```
async function fetchData(url) {
    const response = await fetch(url);
    return response.json();
}
```

- A) `try { ... } catch { ... }`
- B) Return null on error
- C) Ignore errors
- D) Use synchronous fetch

Answer: A) `try { ... } catch { ... }`

QUESTIONS

What might Copilot suggest for this incomplete function that calculates a factorial?

```
function factorial(n) {
    if (n === 0) return 1;
    // implementation missing
}
```

- A) `return n * factorial(n - 1);`
- B) `return n;`
- C) `return 0;`
- D) Leave it incomplete

Answer: A) `return n * factorial(n - 1);`

QUESTIONS

What completion might Copilot provide for a method to filter out odd numbers?

```
function filterOddNumbers(arr) {
    // return array of even numbers
}
```

- A) `return arr.filter(num => num % 2 === 0);`
- B) `return arr.filter(num => num % 2 !== 0);`
- C) `return [];`
- D) Return the input array

Answer: A) `return arr.filter(num => num % 2 === 0);`

QUESTIONS

What might Copilot suggest as a refactoring for this lengthy function?

```
function processInput(input) {
    // multiple operations on input
    return result;
}
```

- A) Split it into smaller helper functions
- B) Write it all in one line
- C) Remove unnecessary comments
- D) Make it synchronous

Answer: A) Split it into smaller helper functions

QUESTIONS

For the following comment, what code might Copilot generate?

```
// Function to check if a string is a palindrome
function isPalindrome(str) {
    // implementation here
}
```

- A) `return str === str.split('').reverse().join('');`
- B) `return str.length % 2 === 0;`
- C) `return str.toLowerCase();`
- D) `return false;`

Answer: A) `return str === str.split('').reverse().join('');`

QUESTIONS

What could be a suggestion from Copilot to improve this function's performance?

```
function findMax(arr) {
    let max = arr[0];
    arr.forEach(num => {
        if (num > max) {
            max = num;
        }
    });
    return max;
}
```

- A) Use `reduce` to find the maximum
- B) Sort the array first
- C) Return a hardcoded value
- D) Use a for loop instead of forEach

Answer: A) Use `reduce` to find the maximum

QUESTIONS

- Write a function that returns the square of a number. Use a comment to guide Copilot to suggest a specific implementation.

```
// Function to calculate the square of a number
function square(num) {
    // return the square of num
}
```

- **Answer:** Copilot might suggest `return num * num;` or `return Math.pow(num, 2);`.

QUESTIONS

- Write a function that filters out odd numbers from an array. Use a comment to guide Copilot.

```
// Function to filter odd numbers from an array
function filterOddNumbers(arr) {
    // return array of even numbers
}
```

- **Answer:** Copilot might suggest `return arr.filter(num => num % 2 === 0);`.

QUESTIONS

- Create a class with a method to add two numbers.

```
class MathOperations {
    // Method to add two numbers
    add(a, b) {
        // implementation here
    }
}
```

- **Answer:** Copilot might suggest `return a + b;`

QUESTIONS

- Define a function to check if a number is prime, with a comment to guide the implementation.

```
// Function to check if a number is prime
function isPrime(n) {
    // implementation here
}
```

- **Answer:** Copilot might suggest checking divisibility up to the square root of `n`.

QUESTIONS

- Define a method in a class that calculates the area of a rectangle.

```
class Rectangle {
    // Method to calculate area
    area(length, width) {
        // return area of rectangle
    }
}
```

- **Answer:** Copilot might suggest `return length * width;`.

QUESTIONS

- Write a function that generates a random number between two values.

```
// Function to generate a random number between min and max
function getRandom(min, max) {
    // implementation here
}
```

- **Answer:** Copilot might suggest `return Math.random() * (max - min) + min;`.

QUESTIONS

- Refactor a function to improve its readability.

```
function getSum(arr) {
    let sum = 0;
    for (let i = 0; i < arr.length; i++) {
        sum += arr[i];
    }
    return sum;
}
```

- **Answer:** Copilot might suggest using `reduce`: `return arr.reduce((total, num) => total + num, 0);`.

QUESTIONS

- Create a basic Express server. Use comments to guide Copilot.

```
// Import express
const express = require('express');
const app = express();

// Set up a basic route
app.get('/', (req, res) => {
    // return a welcome message
});
```

- **Answer:** Copilot might suggest `res.send('Welcome to the server!');`

QUESTIONS

- Write a template for a simple HTML page.

```
// Function to create a basic HTML structure
function createHTML() {
    // return a string of HTML
}
```

- **Answer:** Copilot might suggest returning a template string with `<!DOCTYPE html>` and `<html>` tags.

QUESTIONS

- Write a function to import and use a utility function from another file.

```
// Import utility function
const { utilityFunction } = require('./utils');

function useUtility() {
    // call the utility function
}
```

- **Answer:** Copilot might suggest calling `utilityFunction()` with appropriate arguments

QUESTIONS

- Write a function that checks if an array includes a specific value.

```
// Function to check if value exists in array
function includesValue(arr, value) {
    // implementation here
}
```

- **Answer:** Copilot might suggest `return arr.includes(value);`

QUESTIONS

- Write a function that catches and logs errors.

```
async function fetchData(url) {
    // fetch data from the URL
}
```

- **Answer:** Copilot might suggest wrapping the fetch call in a try-catch block.

QUESTIONS

- Write a Jest test for a function that adds two numbers.

```
// Function to add two numbers
function add(a, b) {
    return a + b;
}

// Test for add function
test('adds 1 + 2 to equal 3', () => {
    // implementation here
});
```

- **Answer:** Copilot might suggest `expect(add(1, 2)).toBe(3);`.

QUESTIONS

- Write a function that safely parses JSON and handles errors.

```
function safeParse(jsonString) {
    // parse the JSON string
}
```

- **Answer:** Copilot might suggest using `try-catch` to handle `JSON.parse` errors.

QUESTIONS

- Write a function that formats a date string. Use a comment to guide Copilot.

```
// Function to format a date string to 'YYYY-MM-DD'
function formatDate(date) {
    // implementation here
}
```

- **Answer:** Copilot might suggest formatting the date using `toISOString()` and `substring()` methods.