**1. Instalar o VSCode e Extensões Necessárias**

Se ainda não tiver o VSCode instalado, baixe e instale-o em: [VSCode](https://code.visualstudio.com/).

Além disso, é recomendado instalar as seguintes extensões para melhorar a experiência no desenvolvimento com Python e GitHub Copilot:

* **Python** (extensão oficial da Microsoft para Python).
* **GitHub Copilot** (para sugestões de código automáticas).

Para instalar o GitHub Copilot, siga as instruções neste [link](https://docs.github.com/en/copilot/getting-started-with-github-copilot).

**2. Introdução ao Github Copilot**

### ****Exercício 1: Gerando Código a Partir de Comentários****

**Objetivo:** Demonstrar como o Copilot gera código automaticamente com base em descrições textuais.

📝 **Instruções:**

1. Abra um arquivo Python (ou outra linguagem) no VS Code.
2. Escreva o seguinte comentário e veja o que o Copilot sugere:

|  |
| --- |
| *# Escreva uma função que receba uma lista de números e retorne apenas os números pares* |

1. Pressione **Tab** para aceitar a sugestão do Copilot. O código gerado pode ser algo como:

|  |
| --- |
| def filtrar\_pares(lista):  return [num for num in lista if num % 2 == 0]  # Exemplo de uso  numeros = [1, 2, 3, 4, 5, 6]  print(filtrar\_pares(numeros)) # [2, 4, 6] |

🎯 **Objetivo do exercício:**

* Testar diferentes descrições para gerar código automaticamente.
* Explorar variações na estrutura da função (exemplo: usar filter() ao invés de list comprehension)

**Exercício 2: Melhorando Código Ineficiente**

**Objetivo:** Mostrar como o Copilot pode sugerir melhorias para código legado.

📝 **Instruções:**

1. Cole este código em um arquivo Python e peça ao Copilot sugestões de refatoração:

|  |
| --- |
| def calcular\_media(notas):  soma = 0  for nota in notas:  soma += nota  return soma / len(notas)  notas = [7, 8, 9, 10]  print(calcular\_media(notas)) # Esperado: 8.5 |

Pergunte no **Copilot Chat:**

* "Como posso otimizar essa função?"
* "Existe uma maneira mais Pythonica de escrever isso?"

💡 **Sugestão possível do Copilot:**

|  |
| --- |
| def calcular\_media(notas):  return sum(notas) / len(notas) |

🎯 **Objetivo do exercício:**

* Praticar refatoração de código com o Copilot.
* Comparar sugestões e avaliar qual é a melhor em termos de legibilidade e desempenho.

**Exercício 3: Criando Testes Automáticos com Copilot**

**Objetivo:** Demonstrar como o Copilot pode gerar testes unitários para funções existentes.

📝 **Instruções:**

1. Dado o código abaixo, peça ao Copilot para sugerir um teste unitário:

|  |
| --- |
| def somar(a, b):  return a + b |

1. Adicione um comentário pedindo o teste:

|  |
| --- |
| *# Criar um teste unitário para a função somar* |

1. O Copilot pode sugerir automaticamente algo como:

|  |
| --- |
| import unittest  class TestSomar(unittest.TestCase):  def test\_somar(self):  self.assertEqual(somar(2, 3), 5)  self.assertEqual(somar(-1, 1), 0)  self.assertEqual(somar(0, 0), 0)  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

🎯 **Objetivo do exercício:**

* Explorar a geração de testes automatizados.
* Modificar os casos de teste e verificar se o Copilot sugere novas verificações.

**Exercício 4: Interpretando Código Legado**

**Objetivo:** Utilizar o Copilot para entender código obscuro.

📝 **Instruções:**

1. Copie o seguinte código em um editor de texto e peça ao Copilot para explicá-lo:

|  |
| --- |
| def fatorial(n):  return 1 if n == 0 else n \* fatorial(n-1) |

1. Pergunte no **Copilot Chat:**
   * "O que essa função faz?"
   * "Posso melhorá-la de alguma forma?"

💡 **O Copilot pode responder algo como:**  
*"Esta é uma função recursiva para calcular o fatorial de um número. Caso base: se n for 0, retorna 1. Caso contrário, multiplica n pelo fatorial de (n-1). Para evitar erros de recursão profunda, considere uma abordagem iterativa."*

🎯 **Objetivo do exercício:**

* Praticar a explicação de código.
* Comparar a resposta do Copilot com a própria análise do código.

**Exercício 5: Debugging com Copilot**

**Objetivo:** Demonstrar como o Copilot pode ajudar a encontrar e corrigir erros.

📝 **Instruções:**

1. Dado este código com um erro, peça ao Copilot para encontrar o problema:

|  |
| --- |
| def dividir(a, b):  return a / b  print(dividir(10, 0)) # Isso causará um erro! |

1. Pergunte no **Copilot Chat:**
   * "Este código pode gerar um erro?"
   * "Como posso tratá-lo?"

💡 **O Copilot pode sugerir algo como:**

|  |  |
| --- | --- |
| def dividir(a, b):  try:  return a / b  except ZeroDivisionError:  return "Erro: Divisão por zero"  print(dividir(10, 0)) # Retorna "Erro: Divisão por zero" |  |

🎯 **Objetivo do exercício:**

* Identificar erros comuns.
* Implementar boas práticas de tratamento de exceções.