Kezdeti lépések:

1. **Regisztráció:**
   - GitHub Education.
   - Diákigazolvány.
2. **Plugin Telepítés:**
   - Visual Studio Code.
   - GitHub Copilot.
3. **Használat:**
   - Engedélyezés.
   - Kódblokkok választás.
   - Integráció.
4. **Billentyűkombinációk:**
   - Elfogadás, elutasítás.
   - Gyors munkavégzés.

GitHub Copilot

Előnyök:

1. Gyorsaság
2. Hatékonyság
3. Kódminőség
4. Tanulás
5. Időmegtakarítás

Veszélyek:

1. Hibás kód
2. Biztonsági rések
3. Függésveszély
4. Jogviták
5. Kreativitáscsökkenés

1. Interaktív mód

Készíts egy függvényt euklideszi algoritmussal

Accept | Discard | Changed 11 lines

```python
def euklideszi_algoritmus(a, b):
    while b != 0:
        a, b = b, a % b
    return a

# Tesztelés
szam1 = 24
szam2 = 36
eredmeny = euklideszi_algoritmus(szam1, szam2)
print(f"A legnagyobb közös osztó: {eredmeny}")
```

Hatékonyság Növelése

1. **Személyre Szabás:**
   - Gyakori használat és visszajelzés.
2. **Kódminőség Ellenőrzése:**
   - Végső kódellenőrzés és tesztelés.
   - A Copilot generált kódjának minőségi értékelése.
3. **Kód Dokumentálás:**
   - Komentek használata.
   - Jólstrukturált leírások.
4. **Tiszta Kód:**
   - Egyszerűség.
   - Megfelelő formázás.
5. **Nyitva Hagyott Fájlok:**
   - Releváns fájlok nyitva.
6. **Pontos Kifejezések:**
   - Kifejező kérések megfogalmazása.
   - Szűkebb kontextusban való munka.

2. Interaktív felület

```python
def is_prim(n)
```

Javaslatok:

Synthesizing 10/10 solutions

Accept Solution
=======
Suggestion 1

```python
def is_prim(n):
    """Checks if a number is prime."""
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
```

3. Szándékkifejező kommentek

```python
# Függvény, amely megállapítja egy számról, hogy prím-e vagy sem.
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, n // 2 + 1):
        if n % i == 0:
            return False
    return True
```