

Edytor kodu źródłowego

Wprowadzenie

W tym ćwiczeniu poznamy **Visual Studio Code (VSC)** w kontekście programowania. Nauczymy się jak tworzyć i edytować projekty, jak korzystać z rozszerzeń wspierających, jak uruchamiać skrypty za pomocą zintegrowanego terminala, oraz jak debugować aplikacje.

1. Tworzenie projektu Python w Visual Studio Code

- Stwórz folder na projekt
W dowolnej lokalizacji na dysku stwórz folder o nazwie `projekt_python`.
- Otwórz folder w VS Code
W Visual Studio Code wybierz „File” -> „Open Folder” i wskaż utworzony folder `projekt_python`
- Tworzenie nowego pliku Python
Kliknij prawym przyciskiem myszy na folder `projekt_python` w eksploratorze po lewej stronie i wybierz „New File”. Nazwij nowy plik `main.py`
- Dodaj kod do pliku `main.py`
Otwórz plik i wpisz poniższy kod Pythona

```
def powitanie(imie):  
    print(f"Cześć, {imie}!")  
  
if __name__ == "__main__":  
    imie = input("Podaj swoje imię: ")  
    powitanie(imie)
```

- Zapisz plik używając skrótu klawiszowego `Ctrl + S` (Windows/Linux) lub `Cmd + S` (macOS).

2. Instalacja rozszerzeń wspierających Python w VS Code

- Otwórz panel rozszerzeń
Kliknij ikonę „Extensions” (lub użyj skrótu `Ctrl + Shift + X`), aby otworzyć panel z rozszerzeniami.

- Zainstaluj rozszerzenie Python:
 - W polu wyszukiwania wpisz **Python** i zainstaluj oficjalne rozszerzenie „Python” od Microsoft.
 - Rozszerzenie to zapewnia obsługę języka Python, umożliwia uruchamianie skryptów, debugowanie i analizę kodu.
- Zainstaluj rozszerzenie do formatowania kodu:
 - W polu wyszukiwania znajdź i zainstaluj rozszerzenie **autopep8**, które automatycznie formatuje kod Python zgodnie z konwencją PEP 8.
- Uruchomienie formatowania:
 - W otwartym pliku Python użyj skrótu **Shift + Alt + F** (Windows) lub **Shift + Option + F** (macOS), aby sformatować kod za pomocą **autopep8**.

3. Uruchamianie skryptów Python w terminalu VS Code

- Otwórz zintegrowany terminal:
 - Wybierz „Terminal” -> „New Terminal” z górnego menu (lub użyj skrótu **Ctrl + Shift + ~**), aby otworzyć zintegrowany terminal w VS Code.
- Uruchom skrypt Python:
 - W terminalu upewnij się, że znajdujesz się w katalogu **projekt_python**, a następnie uruchom plik **main.py**:

```
python main.py
```

- Interakcja ze skryptem:
 - Wpisz swoje imię, kiedy skrypt poprosi o dane, a następnie zobacz, jak wynik wyświetli się w terminalu.

4. Debugowanie kodu Python w Visual Studio Code

- **Dodaj więcej funkcji do skryptu:**
 - Rozbuduj skrypt **main.py**, aby dodawał dwie liczby podane przez użytkownika:

```
def powitanie(imie):  
    print(f"Cześć, {imie}!")  
  
def dodaj_liczby(a, b):  
    return a + b  
  
if __name__ == "__main__":
```

```
imie = input("Podaj swoje imię: ")
powitanie(imie)

a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))

wynik = dodaj_liczby(a, b)
print(f"Wynik dodawania {a} i {b} to: {wynik}")
```

- Dodanie punktów przerwania:
 - Otwórz plik `main.py` i kliknij na marginesie obok numeru linii, aby dodać punkt przerwania przy funkcji `dodaj_liczby`.
- Uruchomienie debugowania:
 - Kliknij ikonę debugowania po lewej stronie (lub użyj skrótu `Ctrl + Shift + D`).
 - Wybierz „Run and Debug” i skonfiguruj debugowanie dla Pythona, jeśli pojawi się takie okno.
 - Uruchom skrypt w trybie debugowania.
- Analiza krok po kroku:
 - Gdy skrypt zatrzyma się na punkcie przerwania, możesz sprawdzić wartości zmiennych, wykonując kod krok po kroku (funkcje „Step Over”, „Step Into”, „Step Out” dostępne w pasku debugera).

5. Automatyzacja i testowanie kodu

- **Stwórz plik testowy:**
 - W folderze `projekt_python` stwórz nowy plik o nazwie `test_main.py`.
- **Dodaj prosty test do pliku `test_main.py`:**
 - W tym pliku wpisz prosty test do funkcji `dodaj_liczby`:

```
from main import dodaj_liczby

def test_dodaj_liczby():
    assert dodaj_liczby(2, 3) == 5
    assert dodaj_liczby(-1, 1) == 0
    print("Wszystkie testy przeszły.")
```

- **Uruchomienie testów w terminalu:**
 - Otwórz terminal w VS Code i uruchom plik testowy:

```
python test_main.py
```

- **Sprawdzenie wyników testów:**
 - Terminal wyświetli komunikat „Wszystkie testy przeszły.”, jeśli testy zakończyły się powodzeniem.

6. Praca z wirtualnym środowiskiem Python (venv) w Visual Studio Code

Wirtualne środowiska pozwalają izolować zależności projektu, co jest niezwykle przydatne przy pracy nad różnymi projektami z różnymi wersjami bibliotek. Dzięki **venv** możemy pracować w niezależnym środowisku bez wpływu na globalne instalacje Pythona.

- Tworzenie wirtualnego środowiska Python (venv)
 - Otwórz projekt w Visual Studio Code:
 - i. Upewnij się, że masz otwarty folder swojego projektu Python (**projekt_python**).
 - Otwórz zintegrowany terminal:
 - i. Kliknij na „Terminal” -> „New Terminal” w menu (lub użyj skrótu **Ctrl + Shift + ~**).
 - Utwórz wirtualne środowisko:
 - i. W terminalu wpisz poniższe polecenie, aby utworzyć nowe wirtualne środowisko o nazwie **venv**:

```
python -m venv venv # Windows
python3 -m venv venv # Linux
```

To polecenie stworzy katalog **venv** w folderze projektu, w którym znajdą się wszystkie zależności oraz sam Python.

- Aktywacja wirtualnego środowiska:
 - Po utworzeniu środowiska, musisz je aktywować.

```
.\venv\Scripts\activate # Windows
source venv/bin/activate # Linux
```

Po aktywacji terminal powinien zmienić się, wskazując, że pracujesz w wirtualnym środowisku (np. (**venv**) przed ścieżką).

- Instalowanie pakietów w wirtualnym środowisku
 - Zainstaluj wybrane pakiety:
 - i. Teraz, gdy środowisko jest aktywowane, możesz zainstalować wymagane pakiety tylko w ramach tego projektu. Przykład instalacji pakietu requests:

```
pip install requests
```

Ten pakiet zostanie zainstalowany lokalnie w środowisku **venv**, a nie globalnie w systemie.

- **Sprawdź zainstalowane pakiety:**
 - i. Możesz zobaczyć wszystkie zainstalowane pakiety w środowisku, używając:

```
pip freeze
```

To polecenie wypisze listę zainstalowanych pakietów i ich wersji.

- Konfiguracja wirtualnego środowiska w VS Code:
 - Wybór interpretera Python:
 - i. W Visual Studio Code upewnij się, że edytor korzysta z właściwego interpretera Pythona (z wirtualnego środowiska). Kliknij na dolny pasek statusu, gdzie znajduje się ścieżka do aktualnie używanego Pythona (po lewej stronie na dole), a następnie wybierz interpreter z folderu `venv`:
 - ii. Ścieżka dla Windows: `.\venv\Scripts\python.exe`
 - iii. Ścieżka dla Linux/macOS: `./venv/bin/python`
 - Sprawdzenie poprawności środowiska:
 - i. W terminalu upewnij się, że aktywne środowisko działa poprawnie. Możesz to zrobić, uruchamiając poniższe polecenie, które wskaże ścieżkę do aktywnego Pythona:

```
which python    # Linux/macOS
where python    # Windows
```

Powinno ono wskazywać ścieżkę do Pythona w katalogu `venv`.

- Wyłączanie wirtualnego środowiska
Po zakończeniu pracy nad projektem możesz wyłączyć wirtualne środowisko, wpisując w terminalu:

```
deactivate
```

To polecenie zamknie aktywne środowisko i powrócisz do globalnego Pythona.