Zajęcia 1. Wprowadzenie do python

1.1. Wstęp

Laboratorium prowadzone jest na platformach komputerowych Raspberry Pi 5, wyposażonych w system operacyjny **Raspberry Pi OS (64-bit) Desktop**. Aby przystąpić do pracy, należy zalogować się na urządzeniach:

- **login:** lab0x - gdzie x to numer grupy
- **hasło:** student

Po zalogowaniu, aby otworzyć środowisko programistyczne **Visual Studio Code (VSC)**, należy wykonać następujące kroki:

1. Otwórz główne menu (ikona maliny w lewym górnym rogu).
2. Z listy **Programming (Programowanie)** wybierz opcję **Visual Studio Code**.

W oknie głównym powita nas okno powitalne. Po lewej stronie znajdują się rozwijane panele:

- Explorer – zawiera informacje na temat obecnego folderu roboczego.
- Search – umożliwia wyszukiwanie.
- Source Control – powiązanie z programem GitHub.
- Run and Debug – okno debugowania.
- Extensions – panel umożliwiający zarządzanie zainstalowanymi rozszerzeniami.

Przed rozpoczęciem pracy, należy utworzyć folder roboczy użytkownika:

1. Otwórz **Manager plików** (ikona folderu w lewym górnym rogu).
2. Przejdź do folderu **Documents** w katalogu **/home/student**.
3. Stwórz folder o nazwie **Twoje imię i nazwisko**.
4. Wszystkie pliki powinny być umieszczone w tym folderze.

W programie **VSC**:

1. W oknie **File** wybierz opcję **Open Folder...**.
2. Wybierz stworzony folder i zaznacz opcję **Yes, I trust the authors**.

1.2. Tworzenie nowego pliku

W panelu **Explorer**:

1. Kliknij prawym przyciskiem myszy na puste miejsce.
2. Wybierz opcję **New File…**.
3. Nadaj nazwę nowemu plikowi, pamiętając o rozszerzeniu **.py** (np. **zadanie_1.py**).

1.3. Uruchamianie programu

Aby uruchomić stworzony program:

1. Z menu **Run** wybierz polecenie **Run Without Debugging** (skrót klawiszowy **Ctrl+F5**).

2. Jeśli jest to pierwszy raz, możesz zostać poproszony o wybór środowiska debugowania. Wybierz **Python Debugger**.

Rezultat działania programu będzie widoczny w oknie **Terminal**. Jeśli terminal nie jest widoczny, możesz go wyświetlić wybierając opcję **Terminal** w menu **View**.

W panelu **View** dostępne są również okna:

- **Problems** – wyświetlające błędy w programie.
- **Output** – komunikaty systemowe i logi związane z działaniem edytora.
- **Debug Console** – komunikaty związane z procesem debugowania kodu.

1.4. Debugowanie programu

Aby rozpocząć debugowanie:

1. Z menu **Run** wybierz **Start Debugging** (lub użyj skrótu **F5**).
2. Po uruchomieniu programu w trybie debugowania, na górze pojawi się pasek narzędzi debugowania, który zawiera następujące opcje:

- **Continue (F5)** – Kontynuuje wykonywanie programu aż do kolejnego punktu przerwania.
- **Step Over (F10)** – Przechodzi do następnej linii kodu, pomijając wywołania funkcji.
- **Step Into (F11)** – Wchodzi w funkcję, która jest wywoływana w danej linii.
- **Step Out (Shift+F11)** – Wychodzi z bieżącej funkcji i wraca do miejsca, które ją wywołało.

---

2. Opracowanie prostych programów w języku Python

Każdy zadanie należy umieścić w osobnym pliku .py.

1. Napisz kod, który pozwala wprowadzić dowolną liczbę całkowitą z klawiatury i wypisać ją na ekran. Zmodyfikuj kod, aby wyświetliła się informacja, czy wczytana liczba jest parzysta.
2. Napisz program, który rozwiąże równanie kwadratowe `ax² + bx + c = 0`. Współczynniki równania mają być wprowadzane z klawiatury.
3. Napisz program, który:
   - Pobiera od użytkownika liczbę **N**, oznaczającą ilość elementów w tablicy.
   - Wczytuje **N** liczb całkowitych od użytkownika i zapisuje je w liście.
   - Wykonuje następujące operacje na tablicy:
     - Wypisuje wprowadzone liczby w kolejności ich podania.
     - Wypisuje te same liczby w odwrotnej kolejności.
     - Oblicza sumę i średnią arytmetyczną elementów tablicy.
     - Wypisuje największą i najmniejszą wartość z tablicy.
   - Przydatne funkcje :
     - `liczby = []` - stworzenie pustej listy o nazwie `liczby`,
     - `liczby.append(liczba)` – dodanie elementu `liczba` na koniec tablicy,
     - `liczby.pop(0)` – usunięcie pierwszego elementu tablicy.

3. Sterowanie GPIO – Migająca dioda LED

Napisz program w Pythonie, który wykorzystuje bibliotekę **gpiozero** do sterowania diodami LED podłączoną do Raspberry Pi. Program powinien sterować trzema diodami LED (czerwoną, żółtą i zieloną), które będą migać zgodnie z typowym cyklem świateł drogowych.

- Numery pinów GPIO (ang. General Purpose Input/Output) podłączonych do diod led znajdują się na płytce: GPIO22 – zielona, GPIO23 – żółta, GPIO24 - czerwona.
- Sekwencja diod powinna być zgodna z typowym cyklem świateł drogowych.
- Zapewnij możliwość zatrzymania programu (poprzez **KeyboardInterrupt**).
- Przydatne funkcje :
    - `from gpiozero import LED` – zaimportowanie klasy `LED` z modułu `gpiozero`,
    - `zielone = LED(22)` – utworzenie obiektu klasy `LED`, przypisując mu pin **GPIO22**,
    - `zielone.on()` – włącza diodę (przyłożenie napięcia do pinu GPIO).
    - `zielone.off()` – wyłącza diodę.
    - `from time import sleep` – zaimportowanie funkcji `sleep` z modułu `time`,
    - `sleep(2)` – wstrzymanie działania programu na określoną liczbę sekund.
- Dodaj do programu obsługę przycisku dla pieszych. Po jego naciśnięciu światła przechodzą w tryb czerwonego światła (na 10 sekund). Następnie wracają do normalnej sekwencji świateł drogowych.
- Przydatne funkcje :
    - `from gpiozero import Button` – zaimportowanie klasy `Button` z modułu `gpiozero`,
    - `przycisk = Button(25)` – utworzenie obiektu klasy `Button`, przypisując mu pin **GPIO25**,
    - `if przycisk.is_pressed:` – instrukcja warunkowa sprawdzająca czy przycisk jest wciśnięty