3. Enumerations (enum)

3.1 What is an Enumeration?

An **enumeration** is a user-defined data type consisting of a set of named integer constants. Enums make code more readable by replacing "magic numbers" with meaningful names.

Python vs C Comparison:

Python	C
No built-in enum (uses constants)	Has `enum` keyword
`RED = 0; GREEN = 1; BLUE = 2`	`enum Color {RED, GREEN, BLUE};`

3.2 Declaring an Enum

Basic Syntax:

```
enum enum_name {
    constant1,
    constant2,
    constant3
};
```

Example:

```
enum Day {
    SUNDAY,     // 0
    MONDAY,     // 1
    TUESDAY,    // 2
    WEDNESDAY,  // 3
    THURSDAY,   // 4
    FRIDAY,     // 5
    SATURDAY    // 6
};
```

Key Points:

- By default, the first constant is assigned value 0
- Each subsequent constant is incremented by 1
- You can explicitly assign values

3.3 Custom Values in Enums

```
enum Status {
    SUCCESS = 1,
    FAILURE = 0,
    PENDING = -1
};

enum Month {
    JANUARY = 1,
    FEBRUARY,    // 2
    MARCH,       // 3
    APRIL,       // 4
    MAY,         // 5
    JUNE,        // 6
    JULY,        // 7
    AUGUST,      // 8
    SEPTEMBER,   // 9
    OCTOBER,     // 10
    NOVEMBER,    // 11
    DECEMBER     // 12
};
```

3.4 Using Enums

```
#include <stdio.h>

enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
};

int main() {
    enum Day today = WEDNESDAY;
```

```
    printf("Today is day number: %d\n", today);  // Output: 3

    if (today == WEDNESDAY) {
        printf("It's the middle of the week!\n");
    }

    // Using enum in switch statement
    switch (today) {
        case MONDAY:
            printf("Start of work week\n");
            break;
        case FRIDAY:
            printf("TGIF!\n");
            break;
        case SATURDAY:
        case SUNDAY:
            printf("Weekend!\n");
            break;
        default:
            printf("Regular work day\n");
    }

    return 0;
}
```

3.5 Enums with Structures

Combining enums with structures creates powerful data models:

```
enum Grade {
    GRADE_A = 90,
    GRADE_B = 80,
    GRADE_C = 70,
    GRADE_D = 60,
    GRADE_F = 0
};

enum StudentStatus {
    ACTIVE,
```

```
    GRADUATED,
    SUSPENDED,
    WITHDRAWN
};

struct Student {
    int id;
    char name[50];
    enum Grade grade;
    enum StudentStatus status;
};

int main() {
    struct Student s1 = {
        .id = 12345,
        .name = "Alice",
        .grade = GRADE_A,
        .status = ACTIVE
    };

    printf("Student: %s\n", s1.name);

    if (s1.status == ACTIVE) {
        printf("Status: Active Student\n");
    }

    if (s1.grade >= GRADE_B) {
        printf("Good performance!\n");
    }

    return 0;
}
```

3.6 Practical Example: Traffic Light System

```
#include <stdio.h>

enum TrafficLight {
    RED,
    YELLOW,
```

```
    GREEN
};

void displayLightAction(enum TrafficLight light) {
    switch (light) {
        case RED:
            printf("STOP! Red light is on.\n");
            break;
        case YELLOW:
            printf("CAUTION! Yellow light is on.\n");
            break;
        case GREEN:
            printf("GO! Green light is on.\n");
            break;
        default:
            printf("Invalid light state.\n");
    }
}

int main() {
    enum TrafficLight currentLight = RED;

    printf("Traffic Light Simulation:\n\n");

    for (int i = 0; i < 3; i++) {
        displayLightAction(currentLight);

        // Cycle through lights
        if (currentLight == RED) {
            currentLight = GREEN;
        } else if (currentLight == GREEN) {
            currentLight = YELLOW;
        } else {
            currentLight = RED;
        }

        printf("Waiting...\n\n");
    }

    return 0;
}
```

Revision #1
Created 2025-10-05 14:44:21 UTC by DS
Updated 2025-10-05 14:44:43 UTC by DS