Data type

Enumerated

An enumerated data type is a non-composite data type defined by a given list of all possible values that has an implied order. It contains no reference to other data when it is defined.

Often referred to as an enum, is a user-defined data type that consists of a set of named values. Enums are often used to represent a finite set of related constants or options. While the exact implementation can vary between programming languages, I'll provide an example using Python.

from enum import Enum, auto

# Define an enumerated data type

class Color(Enum):

RED = auto()

GREEN = auto()

BLUE = auto()

# Using the enumerated data type

selected\_color = Color.GREEN

if selected\_color == Color.RED:

print("Selected color is RED")

elif selected\_color == Color.GREEN:

print("Selected color is GREEN")

elif selected\_color == Color.BLUE:

print("Selected color is BLUE")

In this example, we've created an enumerated data type called Color using the Enum class from the enum module.

It defined three values: RED, GREEN, and BLUE, each associated with an automatically generated unique value using the auto() function.

You can use the enum values in comparisons and to represent specific states or options in your program.

Enums provide better readability and help prevent common mistakes that might occur when using plain integers or strings to represent such values.

The numbers assigned when defining an enum data type are often referred to as "enumerators" or "underlying values." They serve a few important purposes:

* Uniqueness: Each enumerator is assigned a unique integer value. This ensures that each value within the enum is distinct and can be easily compared to other enum values.
* Default Values: Enumerators can act as default values. If you don't explicitly assign values to enum members, they will be automatically assigned incremental values starting from zero.
* Serialization: In some cases, enums are used to represent states or options in data that needs to be serialized (e.g., saved to a file or transmitted over a network). The underlying integers can be more efficient to store or transmit than the enum member names.

from enum import Enum

class Weekday(Enum):

MONDAY = 1

TUESDAY = 2

WEDNESDAY = 3

THURSDAY = 4

FRIDAY = 5

SATURDAY = 6

SUNDAY = 7

# Using the Weekday enum

today = Weekday.WEDNESDAY

if today == Weekday.SATURDAY or today == Weekday.SUNDAY:

print("It's the weekend!")

else:

print("It's a weekday.")

In this example, we've defined an enumerated data type called Weekday using the Enum class. Each weekday is assigned an underlying integer value, starting from 1. The program then demonstrates how to use the enum to determine whether the current day is a weekend day or a weekday.

Remember, the underlying values are there for comparison and identification purposes. You can access the underlying value of an enum member using its .value attribute, like Weekday.MONDAY.value.

from enum import Enum

class Weekday(Enum):

MONDAY = 1

TUESDAY = 2

WEDNESDAY = 3

THURSDAY = 4

FRIDAY = 5

SATURDAY = 6

SUNDAY = 7

# Using the Weekday enum and accessing underlying values

today = Weekday.WEDNESDAY

if today.value in (Weekday.SATURDAY.value, Weekday.SUNDAY.value):

print("It's the weekend!")

else:

print("It's a weekday.")

print("Today's underlying value:", today.value)

In this program, we're accessing the underlying value of the today enum member using the .value attribute. We then compare this value to the values of Weekday.SATURDAY and Weekday.SUNDAY to determine whether it's a weekend or a weekday. Finally, we print out the underlying value of the today enum member.

It's a weekday.

Today's underlying value: 3

Please note that the output will vary depending on the day you run the program. If you change the today value to another day of the week, the output might indicate that it's a weekend if the day is Saturday or Sunday, and it will display the underlying value associated with the selected day.

Answer the following questions.

Question 1: What is an enumerated data type in programming?

Question 2: How do you define an enumerated data type in Python?

Question 3: What is the purpose of underlying values in an enumerated data type?

Question 4: How can you access the underlying value of an enum member in Python?

Question 5: Why would you use an enumerated data type instead of using plain integers or strings to represent constants?

Question 6: Can enums have custom methods?

Question 7: How do you compare enum values in a program?

Question 8: What is the difference between using plain integers and enums to represent constants?

Question 9: In Python, how can you loop through all the values of an enumerated data type?

Question 10: How do enums enhance code maintainability?

Write program for the following using enumerated data type

1. Menu Selection: Create a program that displays a menu of options and handles user input.

2. Traffic Light Simulation: Simulate a traffic light with three states: "Red," "Yellow," and "Green."

3. Days Until Weekend: Write a program that calculates the number of days until the next weekend (Saturday or Sunday).

4. Student Grades: Design a program that manages student grades. Use an enum to represent letter grades ("A," "B," etc.) and create functions to calculate average grades and perform grade-related operations.

5. Seasonal Clothing Suggestions: Develop a program that provides clothing suggestions based on the current season. Use an enum to represent the seasons and display appropriate clothing recommendations for each season.

6. Music Playlist: Create a simple music playlist program that allows the user to add, remove, and play songs. Use an enum to represent the playback states ("Play," "Pause," "Stop") and implement the functionality accordingly.

7. Weather Conditions: Write a program that provides weather information based on user input. Use an enum to represent different weather conditions ("Sunny," "Cloudy," "Rainy") and display appropriate responses.

8. Quiz Application: Design a quiz application where questions are organized by categories (e.g., "Science," "History," "Math"). Use an enum to represent the quiz categories and display questions accordingly.

9. Task Priority Manager: Develop a simple task manager program that allows the user to prioritize tasks. Use an enum to represent task priorities ("Low," "Medium," "High") and manage tasks based on their priority levels.

10. Language Translator: Create a program that translates words or phrases between different languages. Use an enum to represent the supported languages and perform translations using a translation library or service.

These program ideas incorporate enumerated data types to enhance code clarity and organization. You can choose one that aligns with your interests and skills, and then expand upon it to create a complete and functional application.

Pointers

A pointer data type is used to reference a memory location. This data type needs to have information about the type of data that will be stored in the memory location.

In pseudocode the type definition has the following structure, in which ^ shows that the type being declared is a pointer and <Typename> is the type of data to be found in the memory location, for example INTEGER or REAL, or any user-defined data type

Using pseudocode for the enumerated data type for days of the week, declare a suitable pointer to use. Set your pointer to point at today.

Remember, you will need to set up the pointer data type and the pointer variable

enum Weekday

MONDAY = 1

TUESDAY = 2

WEDNESDAY = 3

THURSDAY = 4

FRIDAY = 5

SATURDAY = 6

SUNDAY = 7

declare pointer to Weekday as current\_day

set current\_day to Weekday.WEDNESDAY

print "Today is:", current\_day