**AI Day 05 Notes**

**Syed Mansoor ul Hassan Bukhari**

**Detailed Notes on Python's `match` Statement**

The `match` statement in Python, introduced in version 3.10, provides a more powerful and flexible alternative to traditional `if`, `elif`, `else` statements for pattern matching. It allows you to compare a given variable's value against different patterns, executing the corresponding code block for the first pattern that matches.

**Main Entities of `match` Statement**

* **The `match` Keyword:** Initiates the pattern matching process.
* **Case Clauses:** One or more `case` clauses follow the `match` keyword, each containing a pattern to compare against the variable.
* **Expressions for Each Case:** The code block executed when a pattern matches.

**Important Notes**

* Python 3.10 or later is required to use the `match` statement.
* The `match` statement eliminates the need for `switch` statements used in languages like C++.

match variable\_name:

case 'pattern1':

statement1

case 'pattern2':

statement2

case 'pattern3':

...

case 'patternN':

statement

**Example Usage**

**Basic Match Statement Example**

x = int(input("Enter the value of X: "))

match x:

case 0:

print("case is zero")

case 4:

print("case is 4")

case \_:

print(x)

**Pattern Matching with Conditions**

x = int(input("Enter the value of X: "))

match x:

case 0:

print("case is zero")

case 4:

print("case is 4")

case \_ if x != 90:

print(x, "is not 90")

case \_ if x != 80:

print(x, "is not 80")

case \_ if x != 70:

print(x, "is not 70")

case \_: # Working like an else statement here

print(x)

**Combining Patterns with Conditions**

x = 4

match x:

case 0:

print("x is zero")

case 4 if x % 2 == 0:

print("x % 2 == 0 and case is 4")

case \_ if x < 10:

print("x is less than 10")

case \_:

print(x)

**Example with Even and Odd Number Check**

x = int(input("Enter the Number: "))

match x:

case \_ if x % 2 == 0:

print(x, "the number is even")

case \_:

print(x, "The number is odd")

**Implementing a Function to Get the Day of the Week**

Below is an implementation of a function called `get\_day\_of\_week` that takes an integer representing the day of the week (1 for Monday, 2 for Tuesday, ..., 7 for Sunday) and returns the corresponding name of the day using the `match` statement.

def get\_day\_of\_week(day\_number):

match day\_number:

case 1:

return "Monday"

case 2:

return "Tuesday"

case 3:

return "Wednesday"

case 4:

return "Thursday"

case 5:

return "Friday"

case 6:

return "Saturday"

case 7:

return "Sunday"

case \_:

return "Invalid day number"

**Test the function**

print(get\_day\_of\_week(1)) # Output: Monday

print(get\_day\_of\_week(3)) # Output: Wednesday

print(get\_day\_of\_week(8)) # Output: Invalid day number

**In this implementation:**

* + The function `get\_day\_of\_week` uses a `match` statement to check the value of `day\_number`.
  + Each `case` corresponds to a day of the week, returning the name of the day.
  + The default case (`case \_`) handles invalid day numbers, returning "Invalid day number".

This structure provides a clear and concise way to handle multiple conditions, improving readability and maintainability of the code.