PowerShell Course - Day 2: Important Data Types

---

1. Introduction

- **Welcome back**
- **Recap of the previous session**
- **Overview of today's topics**

---

2. Defining a Class in PowerShell

Example: Dog Class

```
class Dog {
    [string]$Name
    [int]$Age
    [string]$Breed

    Dog([string]$name, [int]$age, [string]$breed) {
        $this.Name = $name
        $this.Age = $age
        $this.Breed = $breed
    }

    [void]Bark() {
        Write-Output "$($this.Name) is barking!"
    }

    [void]Fetch([string]$item) {
        Write-Output "$($this.Name) is fetching the $item!"
    }
}

# Create a new Dog object
$myDog = [Dog]::new("Buddy", 3, "Labrador")
$myDog.Bark()
$myDog.Fetch("ball")
```

---

3. Data Types in PowerShell

3.1 String

Description: Represents text data.

Important Methods:

- **ToUpper():** Converts the string to uppercase.
  ```
  $text = "hello world"
  $uppercaseText = $text.ToUpper()
  ```
- **ToLower():** Converts the string to lowercase.
  ```
  $text = "HELLO WORLD"
  $lowercaseText = $text.ToLower()
  ```
- **Contains(string):** Checks if the string contains the specified substring.
  ```
  $text = "hello world"
  $containsHello = $text.Contains("hello")
  ```
- **Replace(oldValue, newValue):** Replaces all occurrences of a specified string with another string.
  ```
  $text = "hello world"
  $newText = $text.Replace("world", "PowerShell")
  ```
- **Split(char[]):** Splits the string into an array of substrings based on a delimiter.
  ```
  $text = "hello world"
  $words = $text.Split(" ")
  ```
- **Trim():** Removes all leading and trailing white-space characters from the string.
  ```
  $text = "  hello world  "
  $trimmedText = $text.Trim()
  ```

3.2 Integer

Description: Represents whole numbers.

Important Methods:

- **ToString():** Converts the integer to its string representation.
  ```
  $number = 42
  $numberAsString = $number.ToString()
  ```

3.3 Boolean

Description: Represents true or false values.

Important Methods:

- **ToString():** Converts the boolean value to its string representation.
  ```
  $isTrue = $true
  $boolAsString = $isTrue.ToString()
  ```

3.4 Array

Description: Represents a fixed-size sequence of elements of the same type.

Important Methods:

- **Length:** Gets the number of elements in the array.
  ```
  $array = @(1, 2, 3, 4, 5)
  $length = $array.Length
  ```
- **Contains(object):** Checks if the array contains the specified element.
  ```
  $array = @(1, 2, 3, 4, 5)
  $containsThree = $array.Contains(3)
  ```
- **Sort():** Sorts the elements in the entire array.
  ```
  $array = @(5, 3, 1, 4, 2)
  [array]::Sort($array)
  ```
- **Reverse():** Reverses the sequence of the elements in the entire array.
  ```
  $array = @(1, 2, 3, 4, 5)
  [array]::Reverse($array)
  ```

3.5 Hashtable

Description: Represents a collection of key/value pairs that are organized based on the hash code of the key.

Important Methods:

- **Add(key, value):** Adds an element with the specified key and value into the hashtable.
  ```
  $hashtable = @{}
  $hashtable.Add("Name", "John")
  ```
- **Remove(key):** Removes the element with the specified key from the hashtable.
  ```
  $hashtable.Remove("Name")
  ```
- **ContainsKey(key):** Checks if the hashtable contains a specific key.
  ```
  $hashtable.ContainsKey("Name")
  ```
- **ContainsValue(value):** Checks if the hashtable contains a specific value.
  ```
  $hashtable.ContainsValue("John")
  ```
- **Keys:** Gets a collection containing the keys in the hashtable.
  ```
  $keys = $hashtable.Keys
  ```
- **Values:** Gets a collection containing the values in the hashtable.
  ```
  $values = $hashtable.Values
  ```

3.6 DateTime

Description: Represents an instant in time, typically expressed as a date and time of day.

Important Methods:

- **Now:** Gets the current date and time.
  ```
  $now = [DateTime]::Now
  ```
- **AddDays(double):** Returns a new DateTime that adds the specified number of days to the value of this instance.
  ```
  $futureDate = $now.AddDays(10)
  ```
- **AddMonths(int):** Returns a new DateTime that adds the specified number of months to the value of this instance.
  ```
  $futureDate = $now.AddMonths(1)
  ```
- **AddYears(int):** Returns a new DateTime that adds the specified number of years to the value of this instance.
  ```
  $futureDate = $now.AddYears(1)
  ```
- **ToString():** Converts the DateTime to its string representation.
  ```
  $dateString = $now.ToString()
  ```

3.7 Custom Objects (PSCustomObject)

Description: Represents a custom object that you can create on-the-fly in PowerShell to store structured data.

Important Methods:

- **New-Object -TypeName PSCustomObject -Property @{}**
  ```
  $person = New-Object -TypeName PSCustomObject -Property @{
      Name = "John"
      Age = 30
      Occupation = "Engineer"
  }
  ```
- **Add-Member -MemberType NoteProperty -Name ”PropertyName” -Value ”Value”**
  ```
  $person | Add-Member -MemberType NoteProperty -Name "Country" -Value "USA"
  ```

---

4. Summary and Q&A

- **Recap of today’s topics**
- **Questions and discussion**
