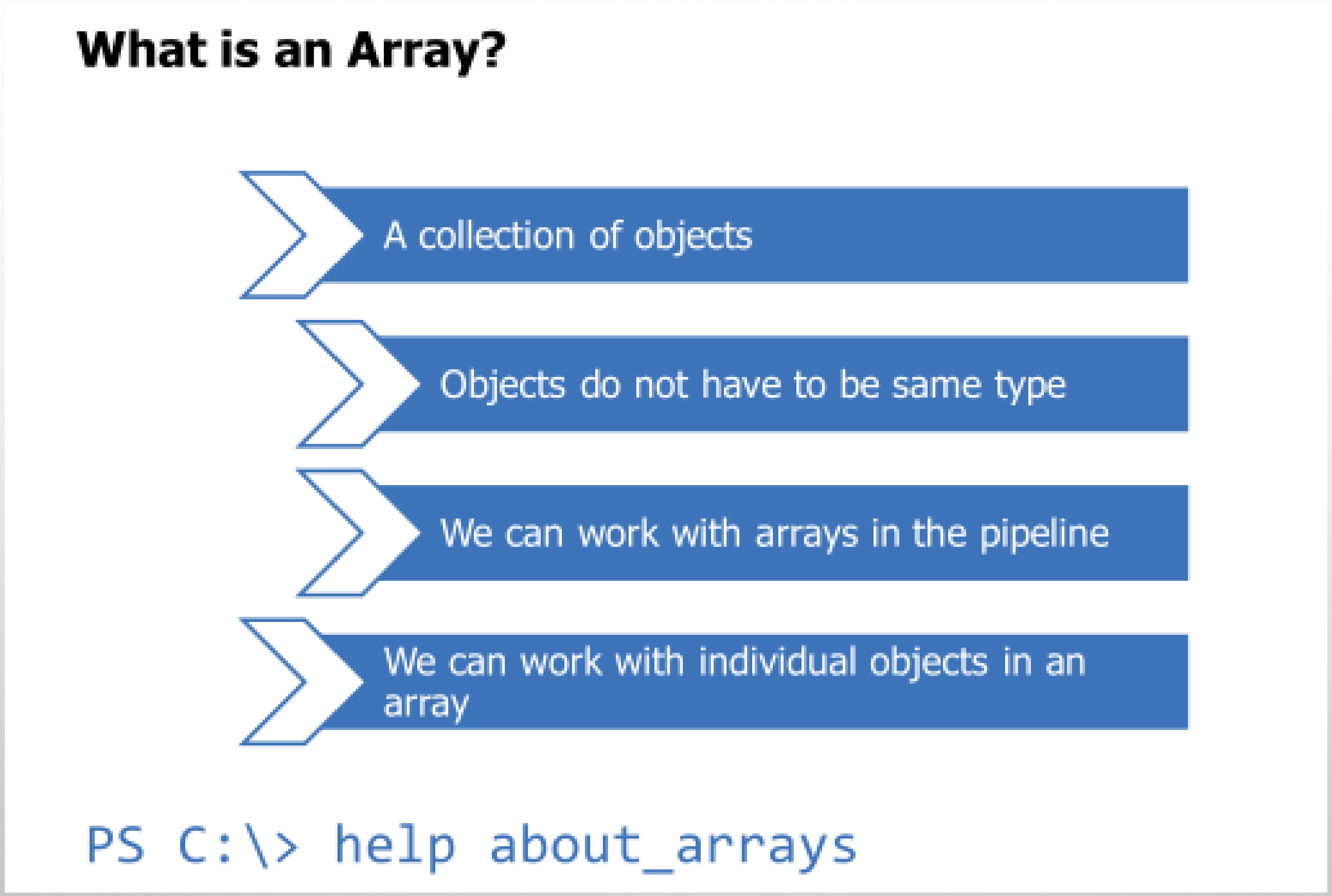
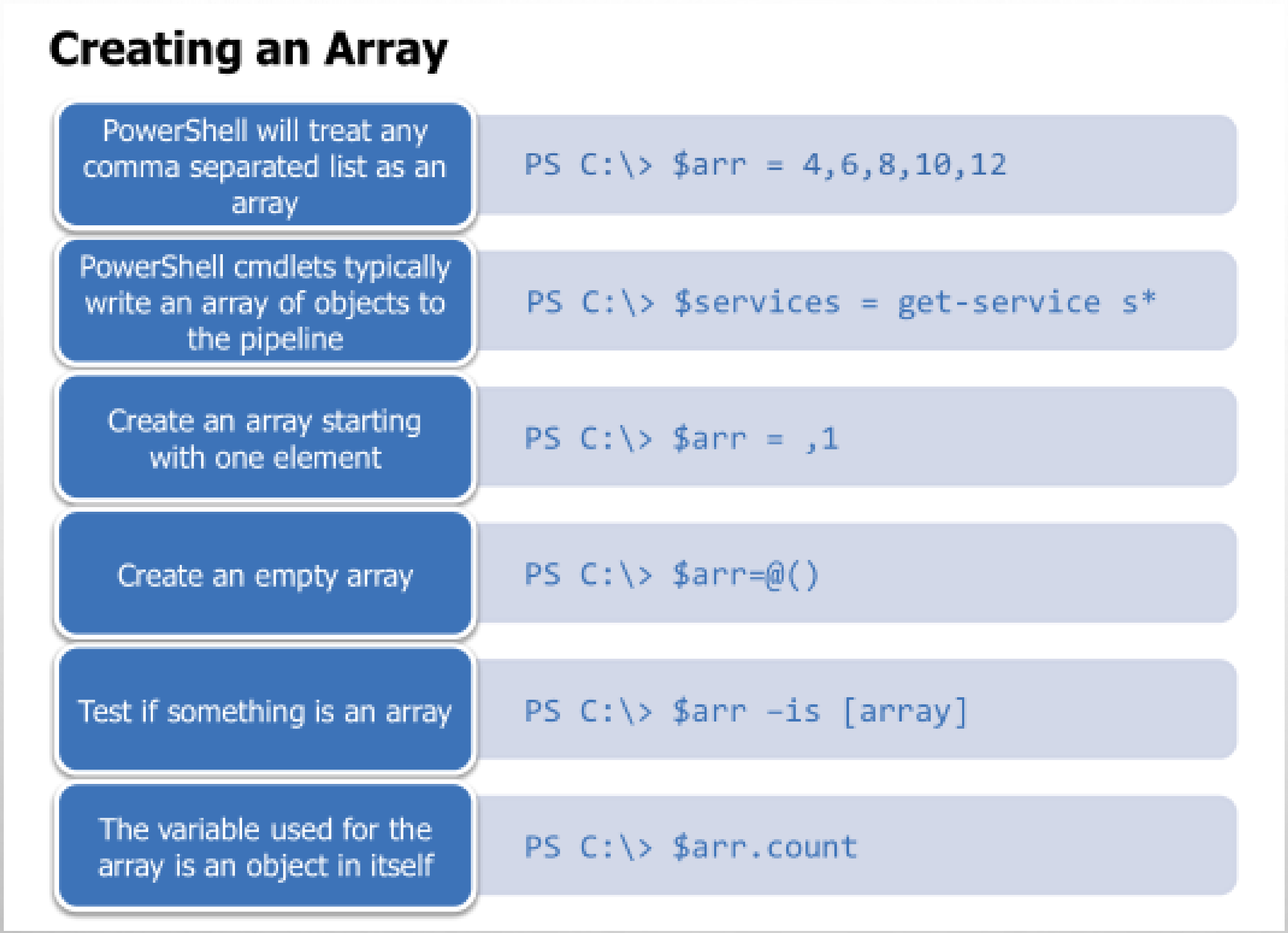
Arrays and Hash Tables



What is an Array?

- A collection of objects
- Objects do not have to be same type
- We can work with arrays in the pipeline
- We can work with individual objects in an array

```
PS C:\> help about_arrays
```



Creating an Array

PowerShell will treat any comma separated list as an array	`PS C:\> $arr = 4,6,8,10,12`
PowerShell cmdlets typically write an array of objects to the pipeline	`PS C:\> $services = get-service s*`
Create an array starting with one element	`PS C:\> $arr = ,1`
Create an empty array	`PS C:\> $arr=@()`
Test if something is an array	`PS C:\> $arr -is [array]`
The variable used for the array is an object in itself	`PS C:\> $arr.count`

Enumerating an Array

Items in arrays are counted starting at 0

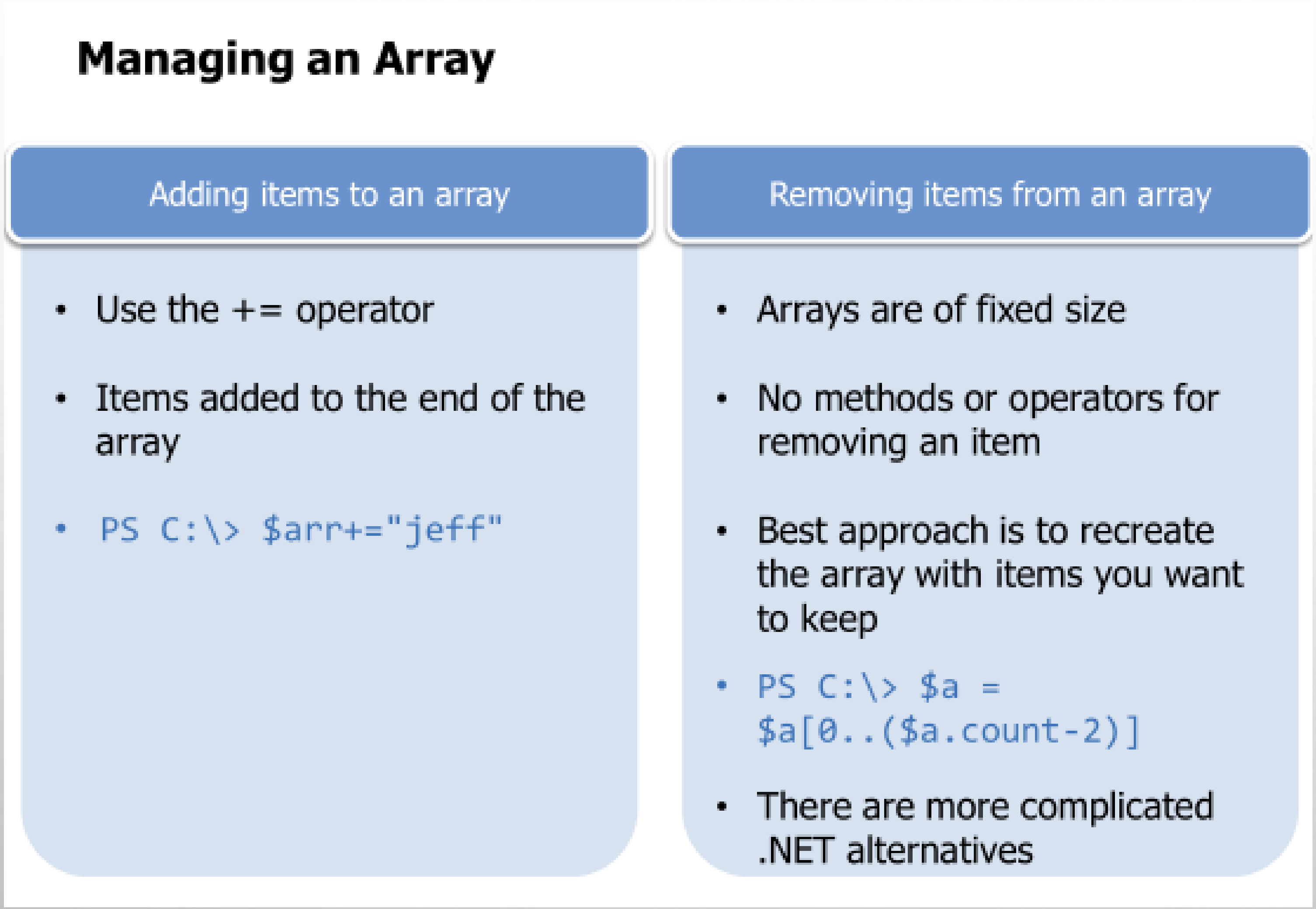
Write the array to the pipeline and PowerShell will automatically enumerate it

Use ForEach

- `PS C:\> foreach ($item in $arr) { $item }`

Use [i] syntax to reference an individual item in an array

- `PS C:\> $s[0]`
- `PS C:\> $s[-1]`
- `PS C:\> $s[-2]`
- `PS C:\> $s[2..4]`
- `PS C:\> $s[-4..-1]`



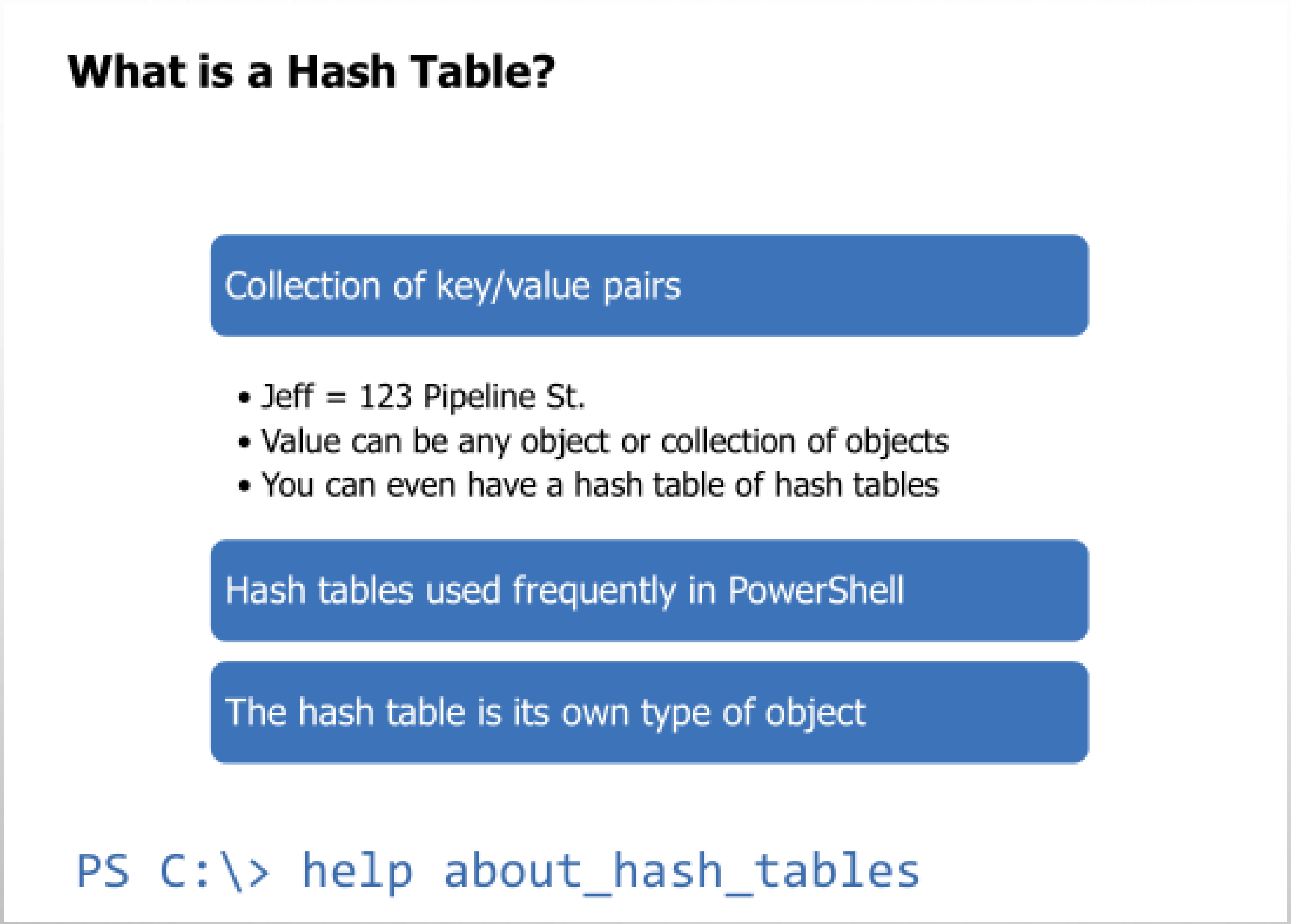
Managing an Array

Adding items to an array

- Use the += operator
- Items added to the end of the array
- `PS C:\> $arr+="jeff"`

Removing items from an array

- Arrays are of fixed size
- No methods or operators for removing an item
- Best approach is to recreate the array with items you want to keep
- `PS C:\> $a = $a[0..($a.count-2)]`
- There are more complicated .NET alternatives



What is a Hash Table?

Collection of key/value pairs

- Jeff = 123 Pipeline St.
- Value can be any object or collection of objects
- You can even have a hash table of hash tables

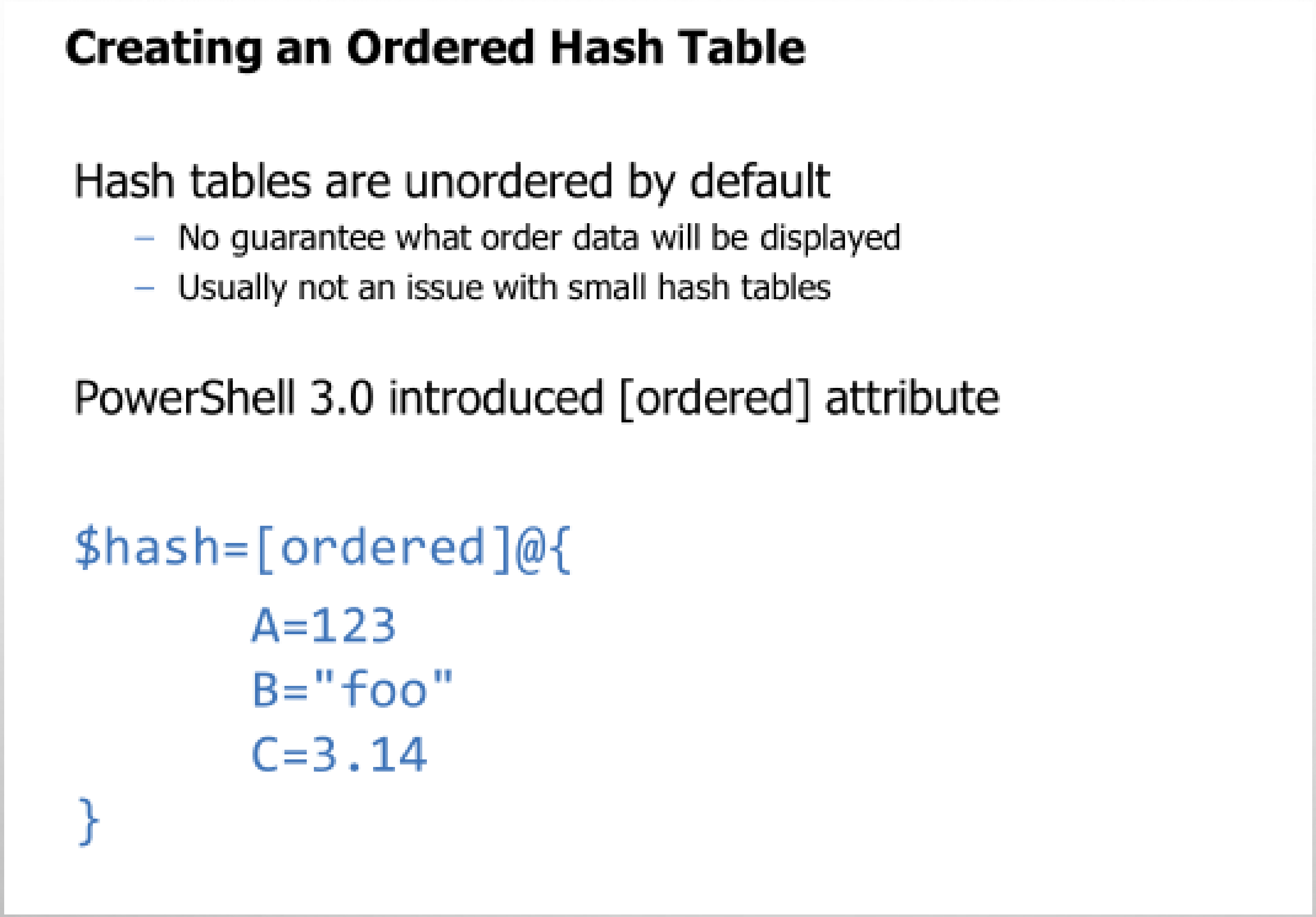
Hash tables used frequently in PowerShell

The hash table is its own type of object

```
PS C:\> help about_hash_tables
```

Creating a Hash Table

- @{Key=Value;Key2=Value;Key3=Value}
- Create an empty hash table
  - `-PS C:\> $hash = @{}`
- Some cmdlets will create hash tables
- You can modify, add, and delete items in a hash table



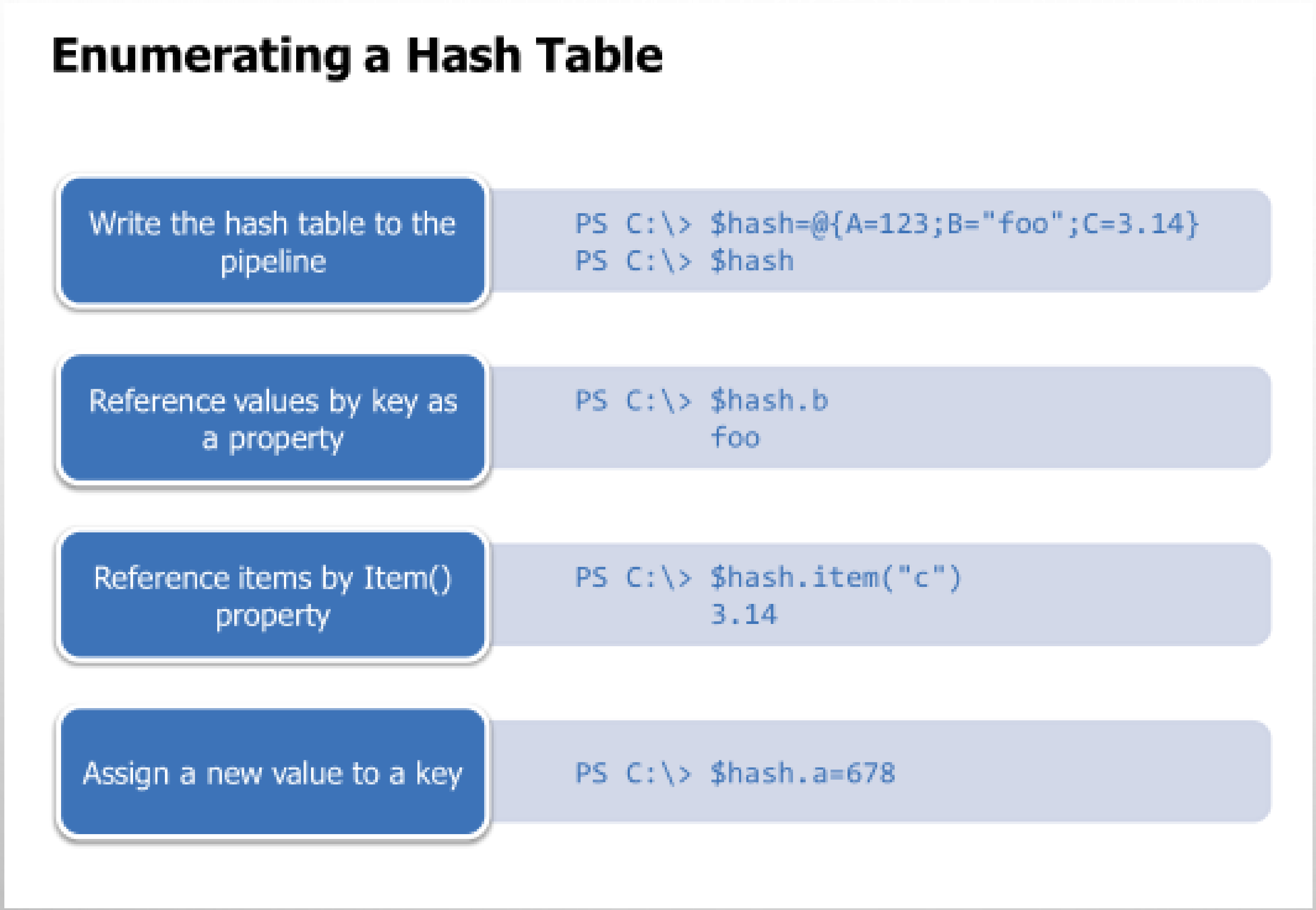
Creating an Ordered Hash Table

Hash tables are unordered by default

- No guarantee what order data will be displayed
- Usually not an issue with small hash tables

PowerShell 3.0 introduced [ordered] attribute

```
$hash=[ordered]@{
      A=123
      B="foo"
      C=3.14
}
```



Enumerating a Hash Table

Write the hash table to the pipeline

```
PS C:\> $hash=@{A=123;B="foo";C=3.14}
PS C:\> $hash
```

Reference values by key as a property

```
PS C:\> $hash.b
        foo
```

Reference items by Item() property

```
PS C:\> $hash.item("c")
        3.14
```

Assign a new value to a key

```
PS C:\> $hash.a=678
```

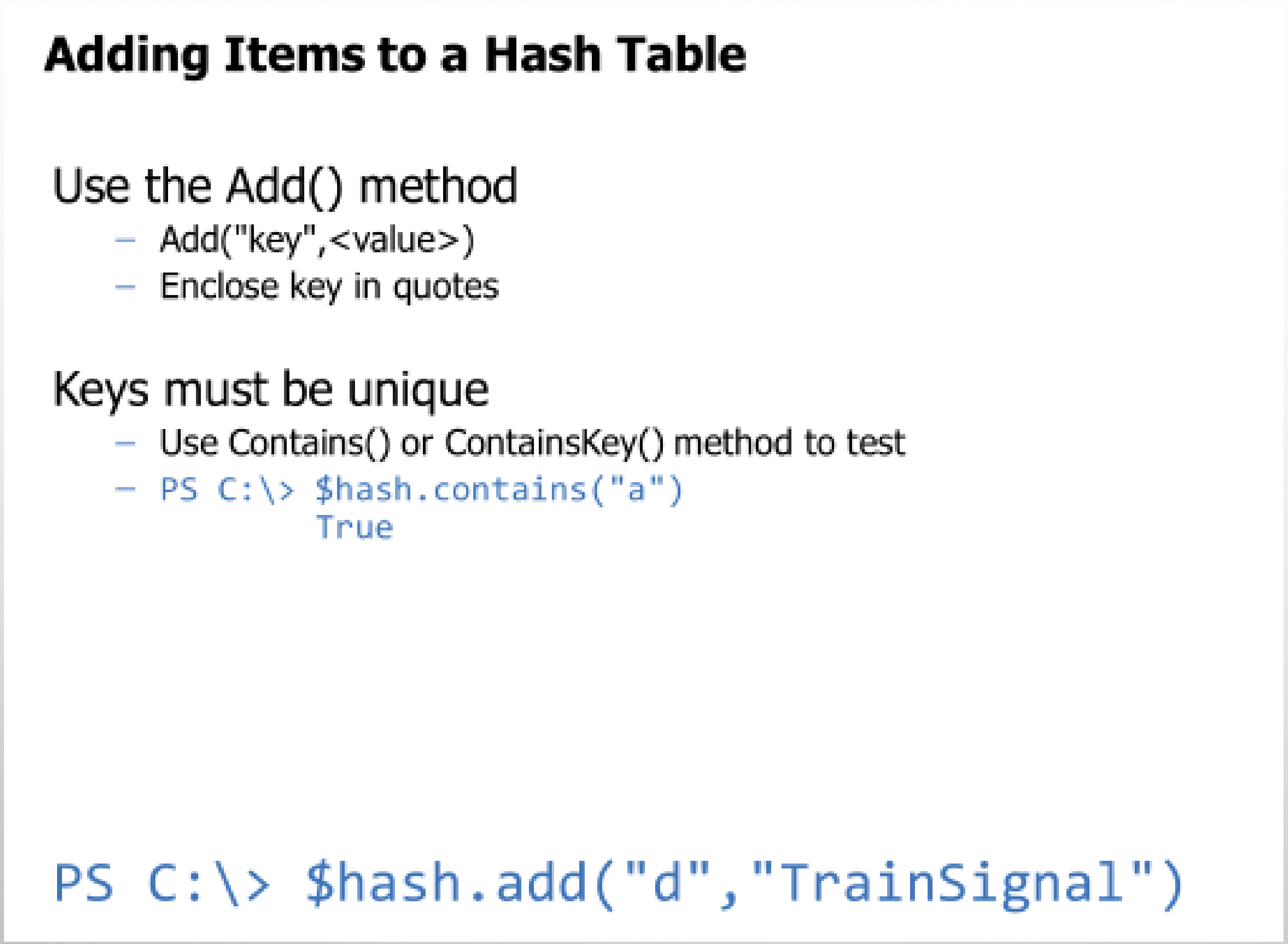
Enumerating a Hash Table

Hash tables cannot be sorted by their keys

Use the GetEnumerator() method

Creates a System.Collections.DictionaryEntry object

```
PS C:\>  $source.GetEnumerator() | where { $_.name -Match
"Windows"}
```



Adding Items to a Hash Table

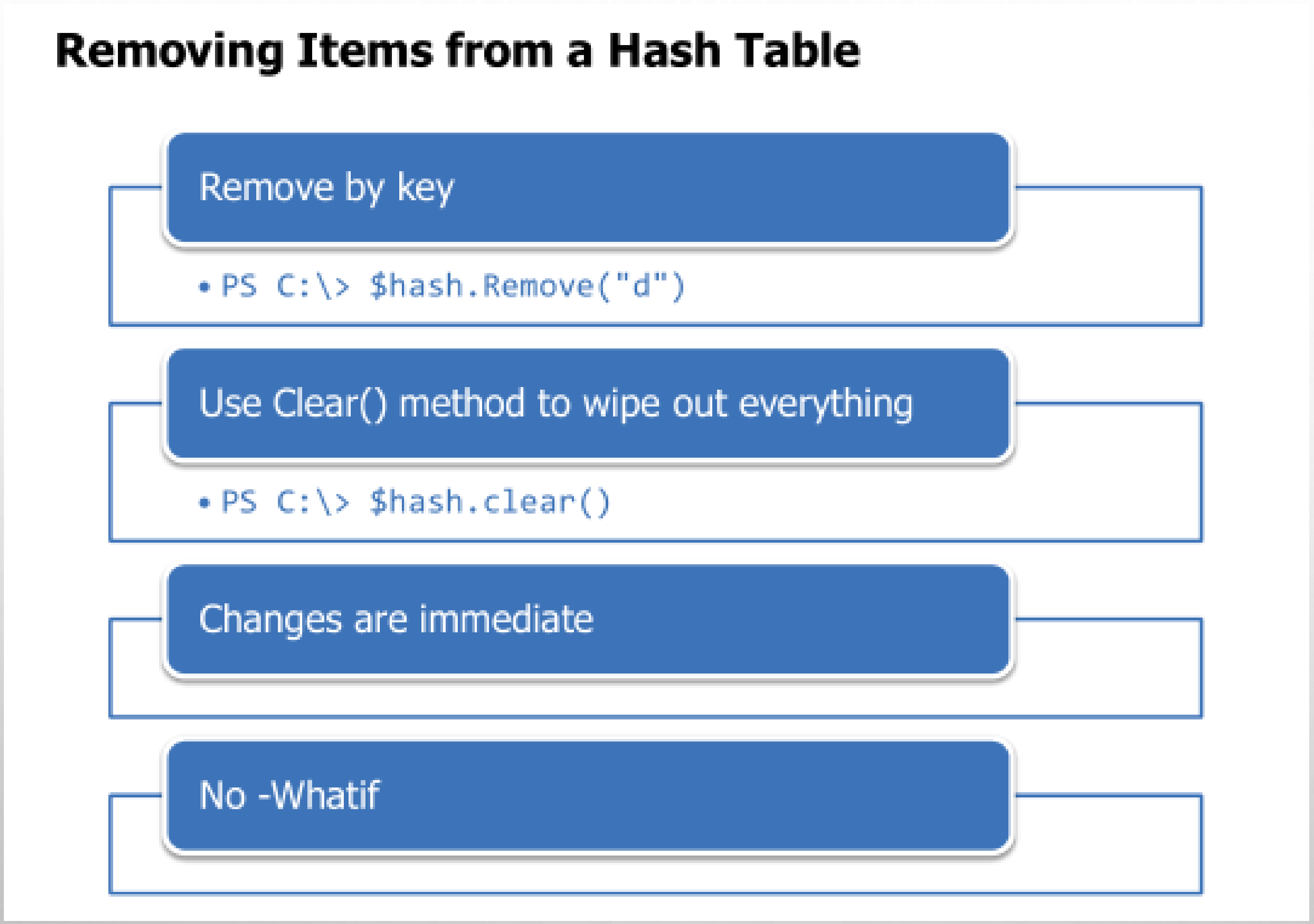
Use the Add() method

- Add("key",<value>)
- Enclose key in quotes

Keys must be unique

- Use Contains() or ContainsKey() method to test
-
```
PS C:\> $hash.contains("a")
        True
```

```
PS C:\> $hash.add("d","TrainSignal")
```



Removing Items from a Hash Table

Remove by key

- `PS C:\> $hash.Remove("d")`

Use Clear() method to wipe out everything

- `PS C:\> $hash.clear()`

Changes are immediate

No -Whatif