Microsoft

PowerShell Flow Control

Module 14

Learnings covered in this Unit

- How to use Selection Commands
- How to use Loop Commands
- How to use Switch Command
- How To Use Flow Control Key Words

PowerShell Sequences, Selections, and Loops

Why Use Sequences, Selections, and Loops

CONTROL THE FLOW OF CODE

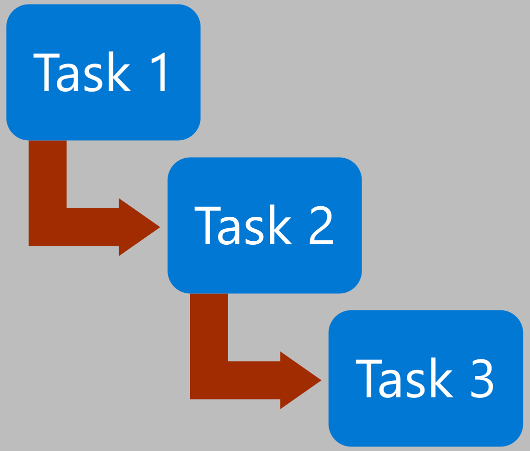
WORK ITERATION OR OBJECT BASED

PROCESS MULTIPLE ITEMS

SOME LOOPS CAN RUN VALIDATIONS BEFORE OR AFTER EXECUTION

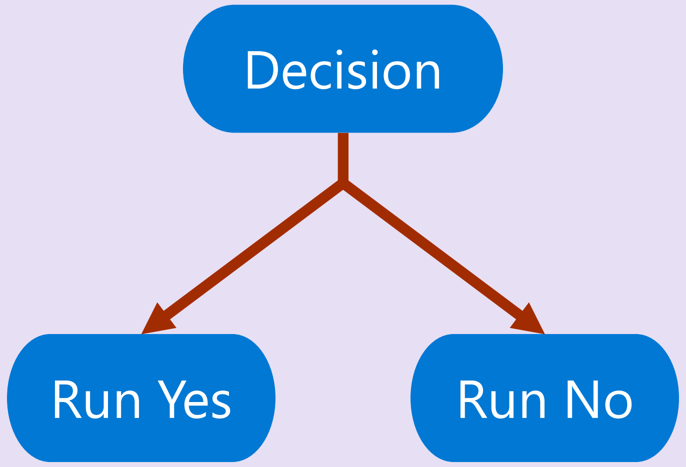
PowerShell's Sequences, Selections and Loops

Sequence



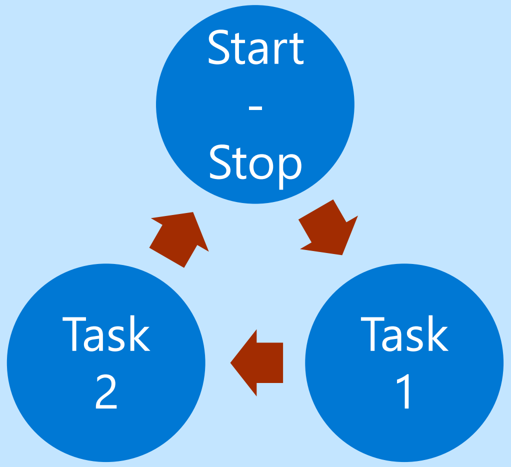
- Script
- Function
- Code Block

Selection

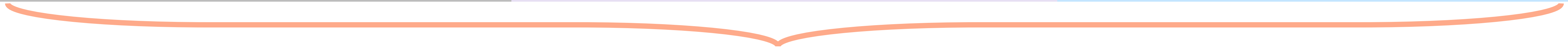


- If
- If Else
- If Elseif
- If Elseif Else

Loop



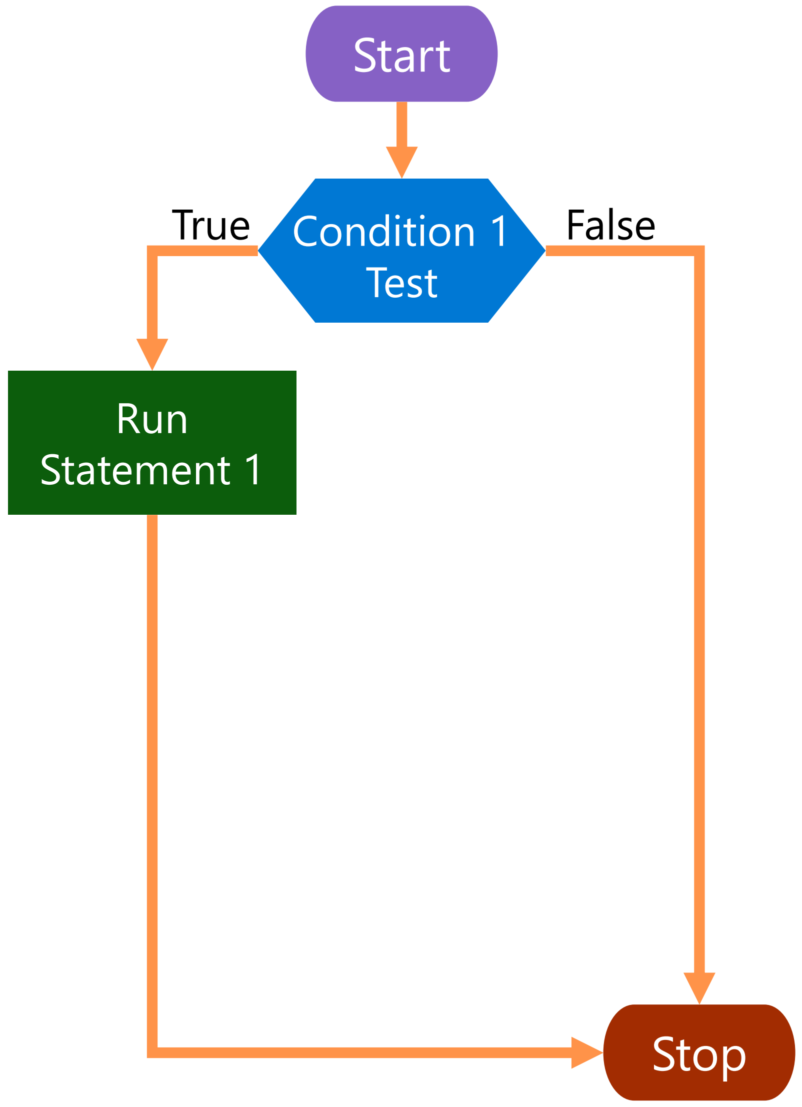
- For
- While
- Foreach
- Do Until
- Do While



Switch

PowerShell Code Selections

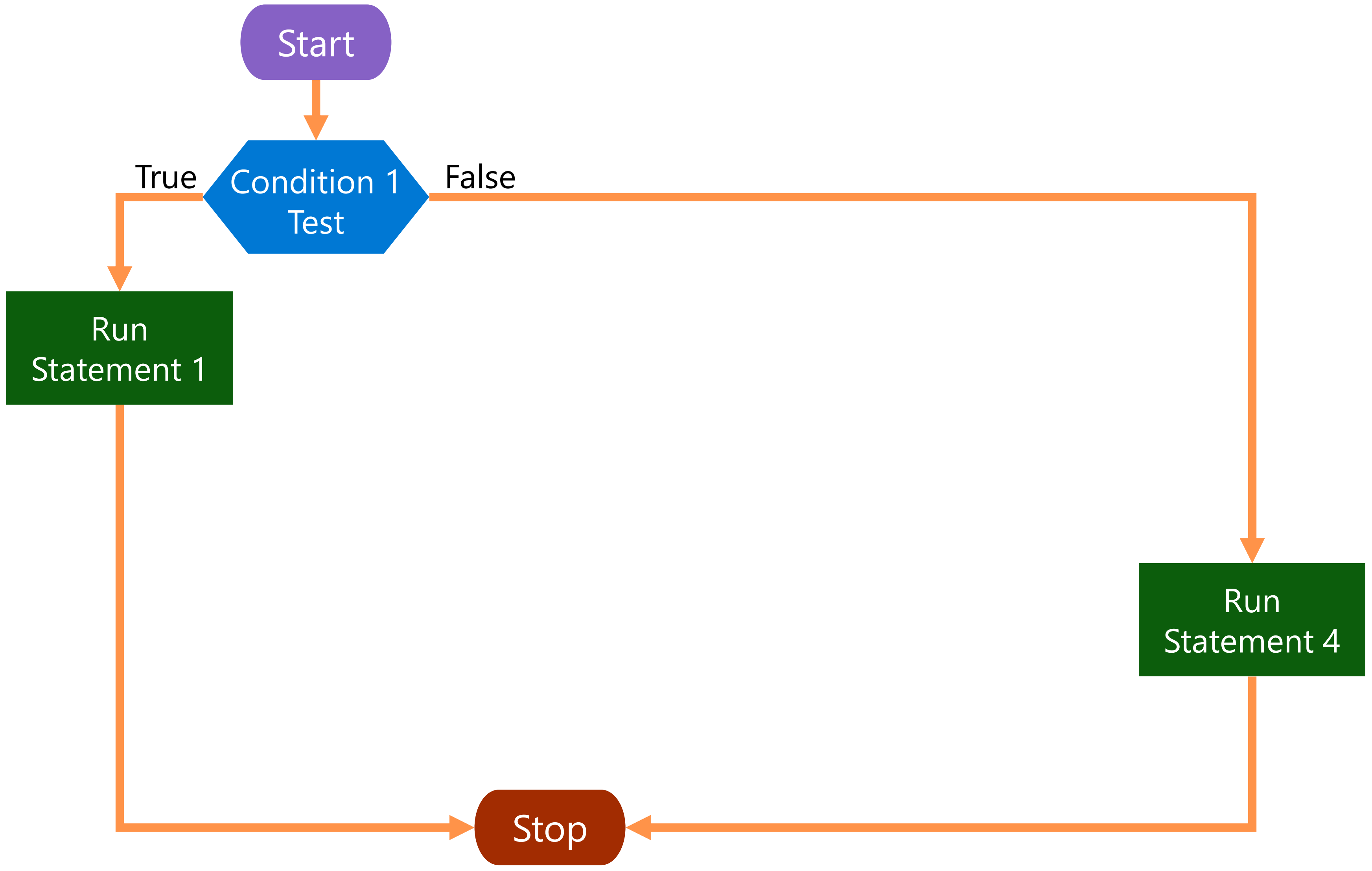
"If" Statement Series of Selection Controls



```
If (Condition 1)
{Statement 1}
```

```
If (1 -eq 2)
{Write-Host "Not 1"}
```

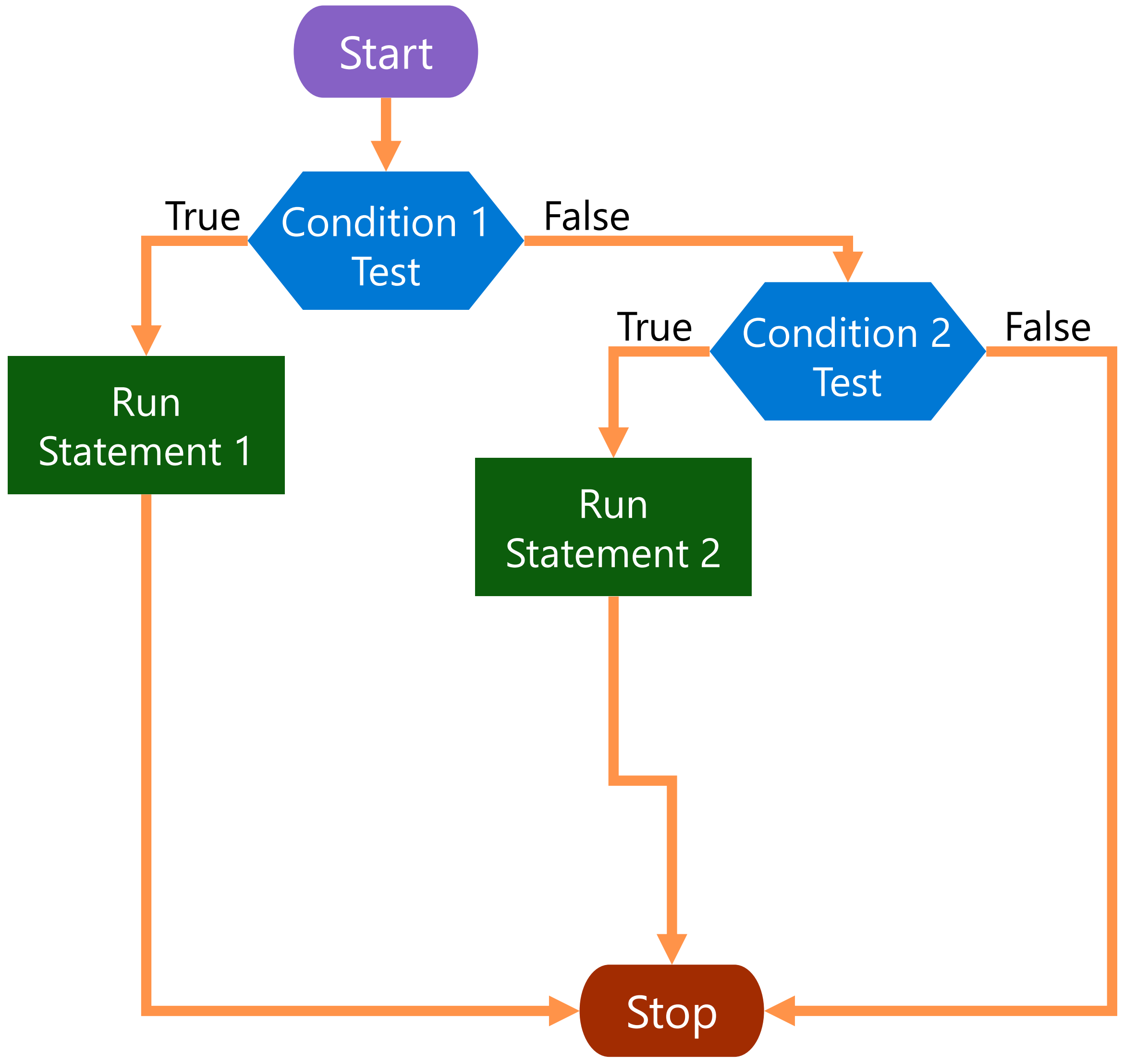
“If” Statement Series of Selection Controls



```
If (Condition 1)
{Statement 1}
Else
{Statement 4}
```

```
If (1 -eq 2)
{Write-Host "Not 1"}
Else
{Write-Host "Hmmmm"}
```

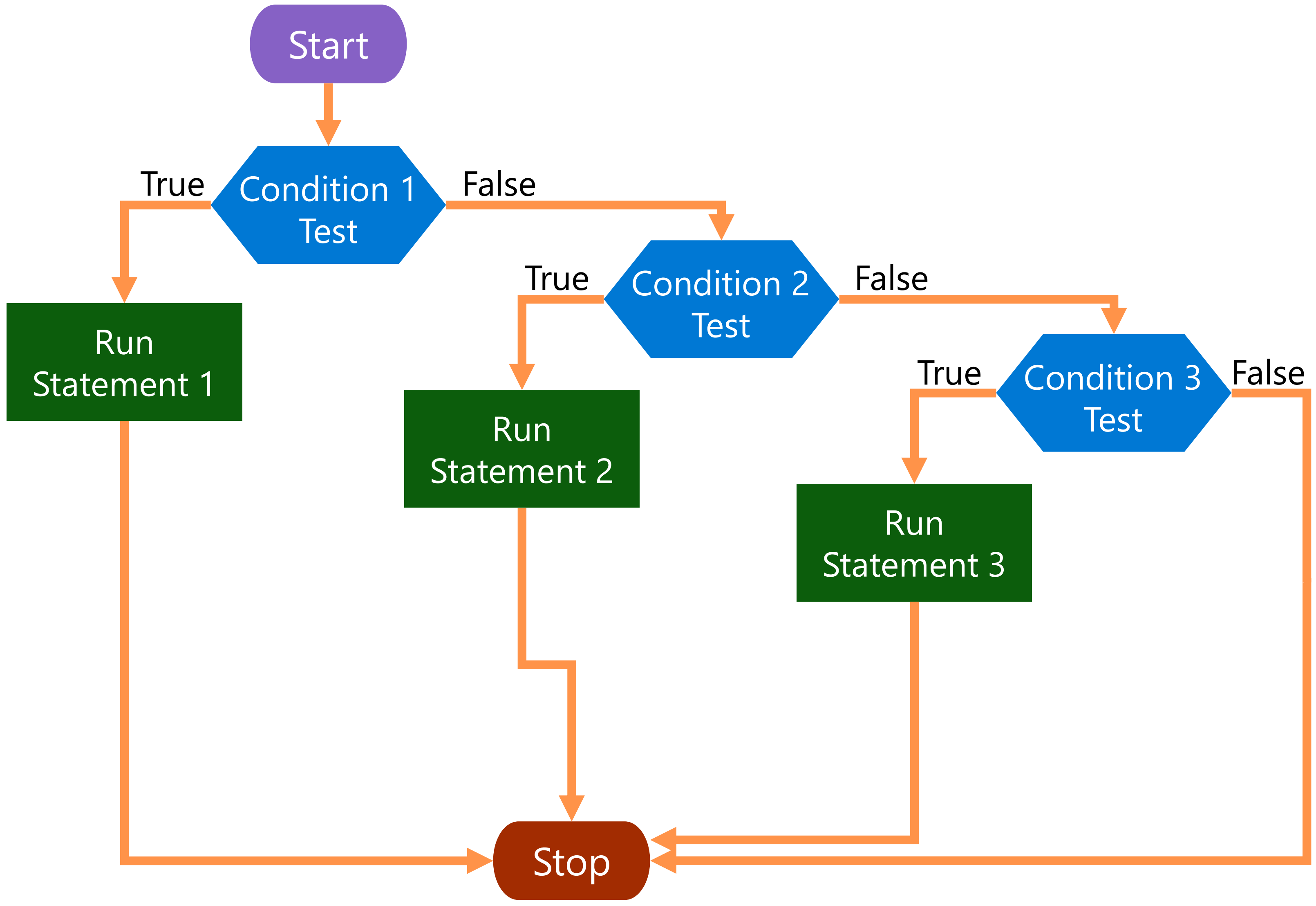
“If” Statement Series of Selection Controls



```
If (Condition 1)
{Statement 1}
Elseif (Condition 2)
{Statement 2}
```

```
If (1 –eq 2)
{Write-Host "Not 1"}
Elseif (1 –eq 3)
{Write-Host "Not 1"}
```

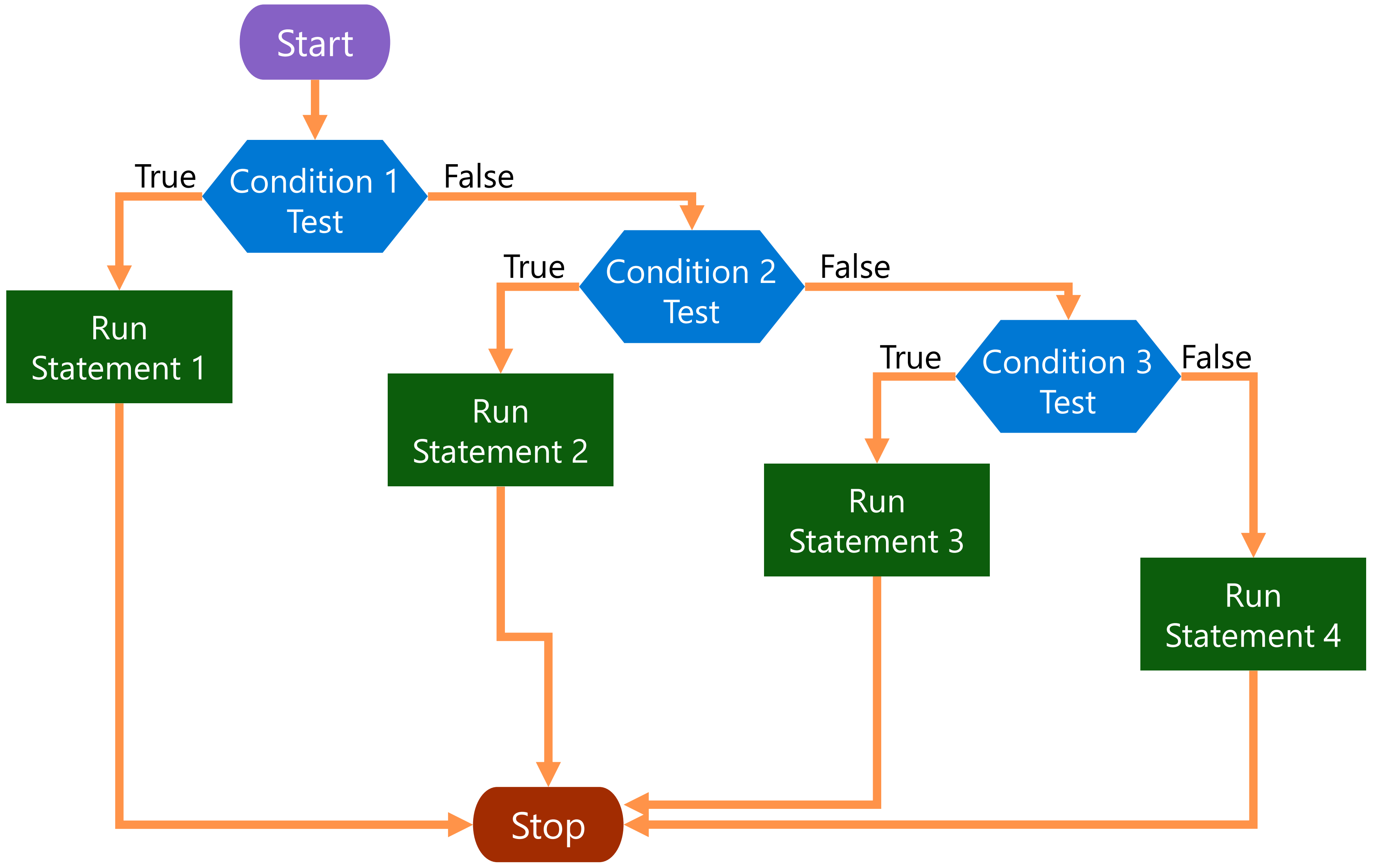
“If” Statement Series of Selection Controls



```
If (Condition 1)
{Statement 1}
Elseif (Condition 2)
{Statement 2}
Elseif (Condition 3)
{Statement 3}
```

```
If (1 –eq 2)
{Write-Host "Not 1"}
Elseif (1 –eq 3)
{Write-Host "Not 1"}
Elseif (1 –eq 4)
{Write-Host "Not 1"}
```

“If” Statement Series of Selection Controls



```
If (Condition 1)
{Statement 1}
Elseif (Condition 2)
{Statement 2}
Elseif (Condition 3)
{Statement 3}
Else
{Statement 4}
```

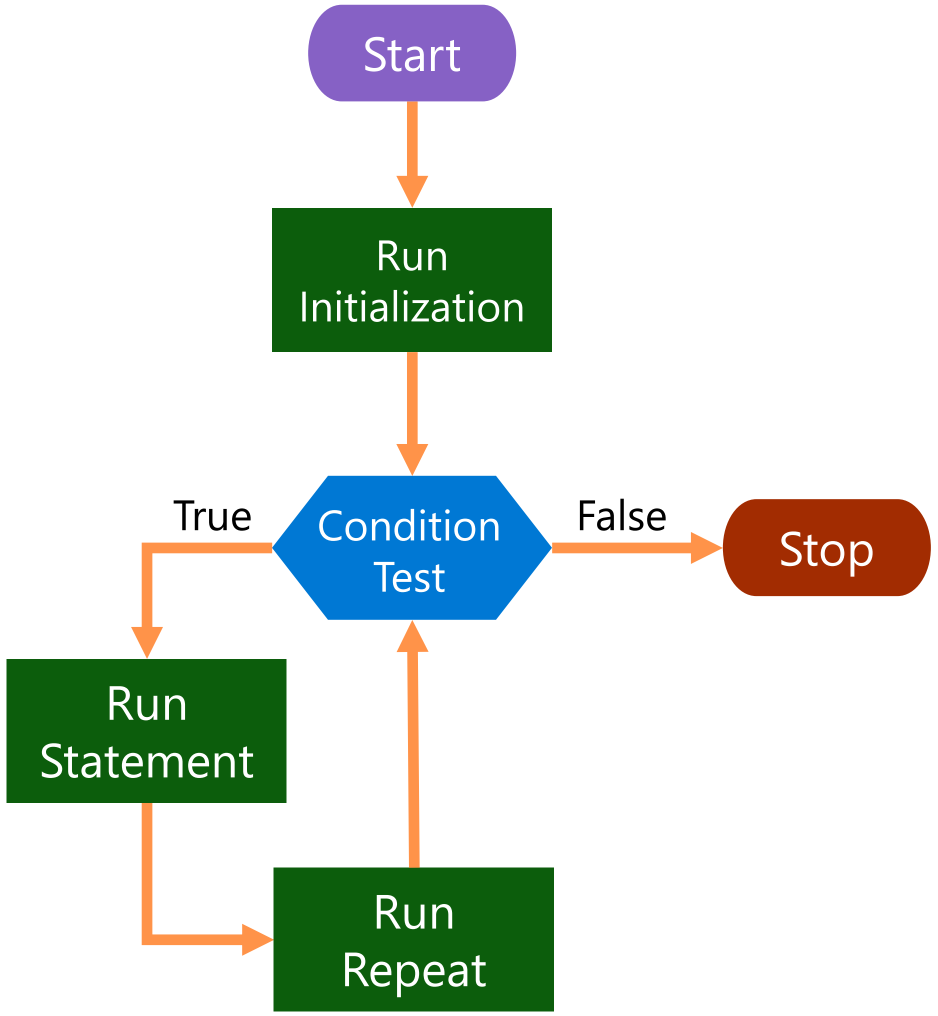
```
If (1 -eq 2)
{Write-Host "Not 1"}
Elseif (1 -eq 3)
{Write-Host "Not 1"}
Elseif (1 -eq 4)
{Write-Host "Not 1"}
Else
{Write-Host "Hmmmm"}
```

Demonstration

If Statements

PowerShell Code Loops

"For" Loop



Initialization:
A **one-time** code block that can perform pre loop code preparations

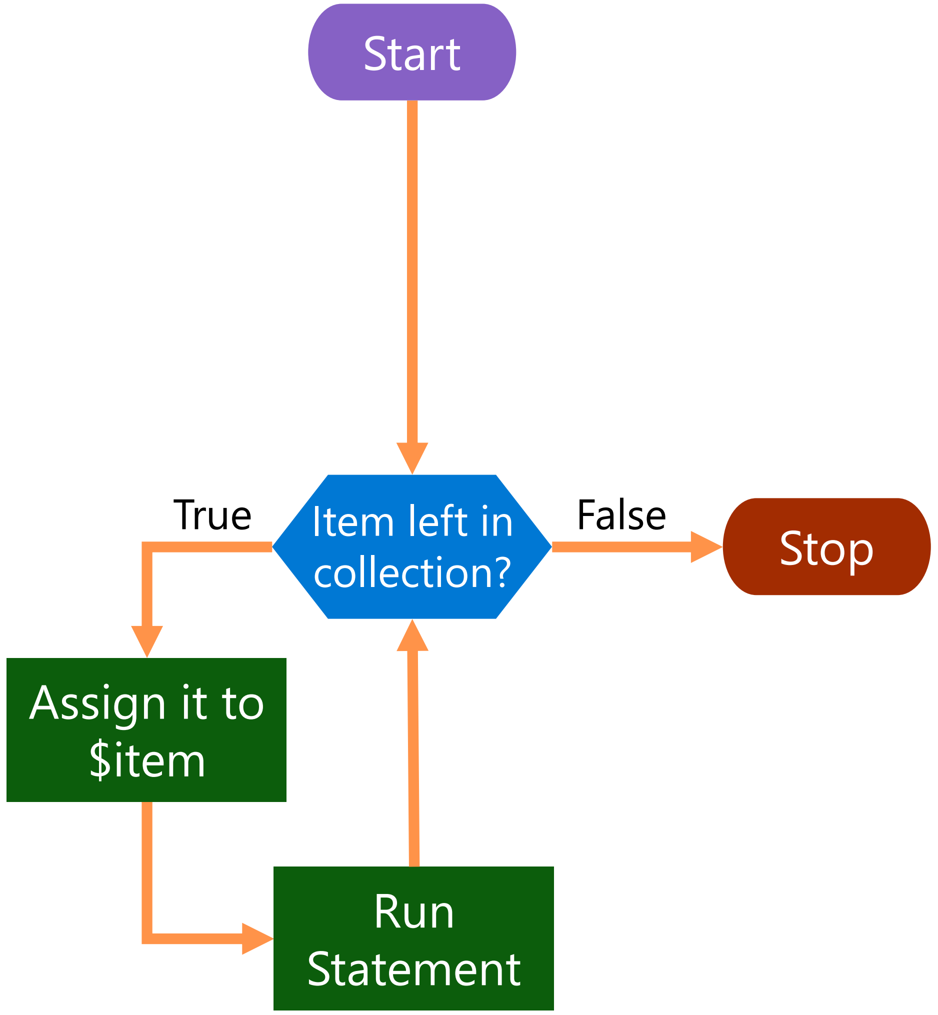
Condition:
Evaluates a comparison statement for a **$true** value

Repeat:
A code block that runs after the statement if the condition is true

```
For (Init; Condition; Repeat){Statement}
```

```
For ($var=5; $var -lt 11; $var++)
{Write-Host "user$var"}

user5
user6
user7
user8
user9
user10
```

“Foreach” Loop



Description:

Works on **each item** in collection and process them until there are no more items to process

```
foreach ($item in $collection)
{Statement}
```

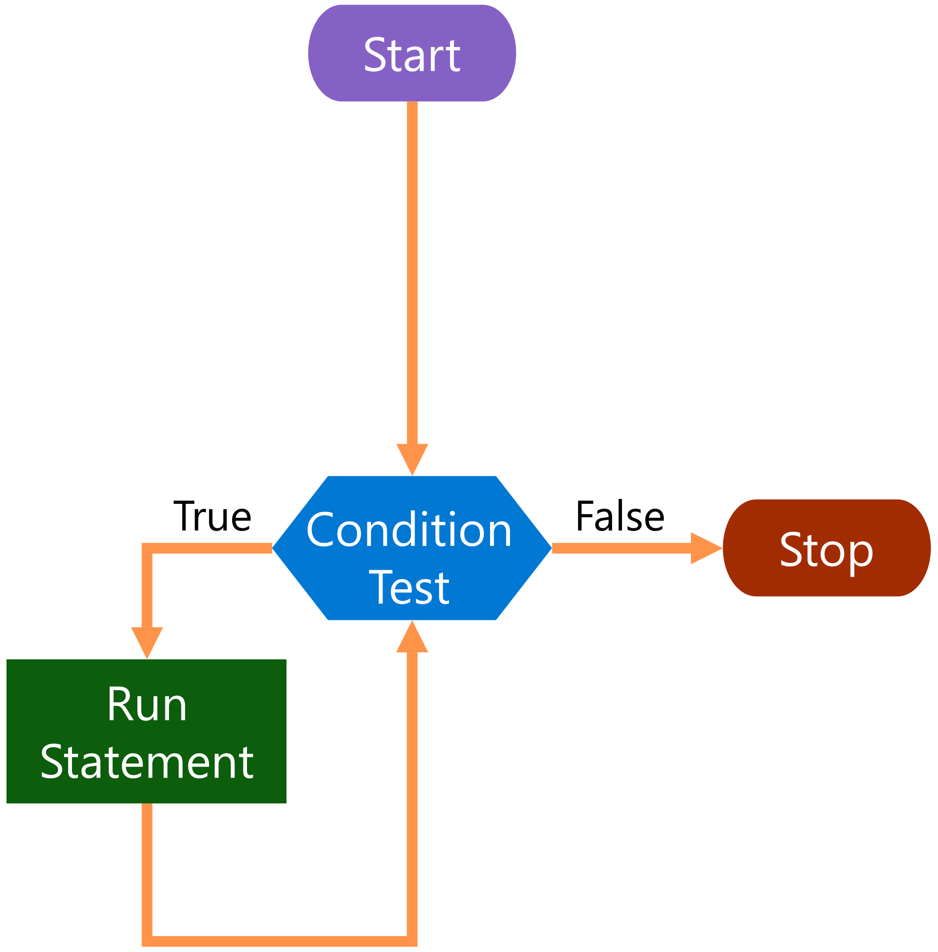
```
$vars = 1, 2, 3

foreach ($var in $vars)
{Write-Host “user$var"}

user1
user2
user3
```

Demonstration

For Loop

Foreach Loop

“While” Loop



Description:

A While loop runs as long as its condition is still **$true**

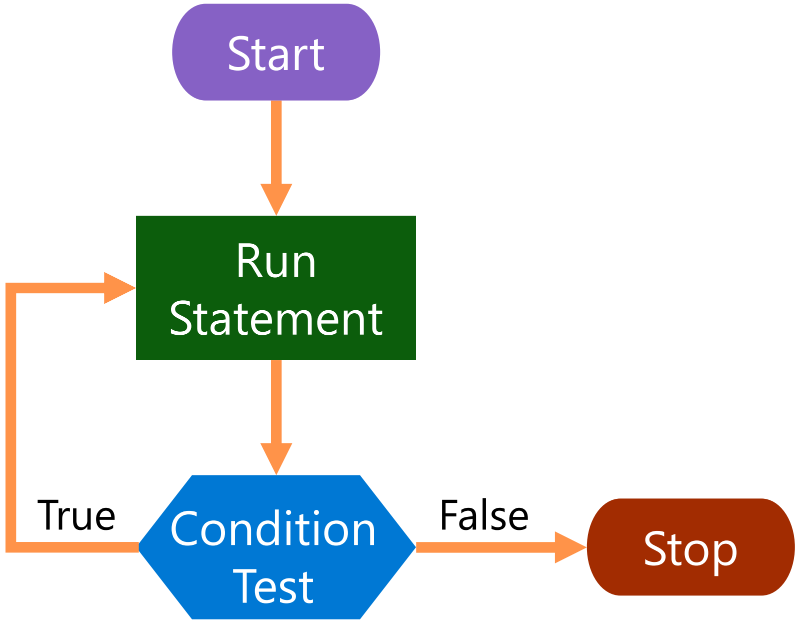
Condition:

Evaluates a comparison statement for a **$true** condition

```
while (Condition) {Statement}
```

```
 $var = 0
 while ($var -lt 3)
{
   Write-Host "user$var"
   $var++
}

user0
user1
user2
```

“Do While” Loop



Description:

A Do While loop **first** runs its statement and then tests if the condition is still **$true**

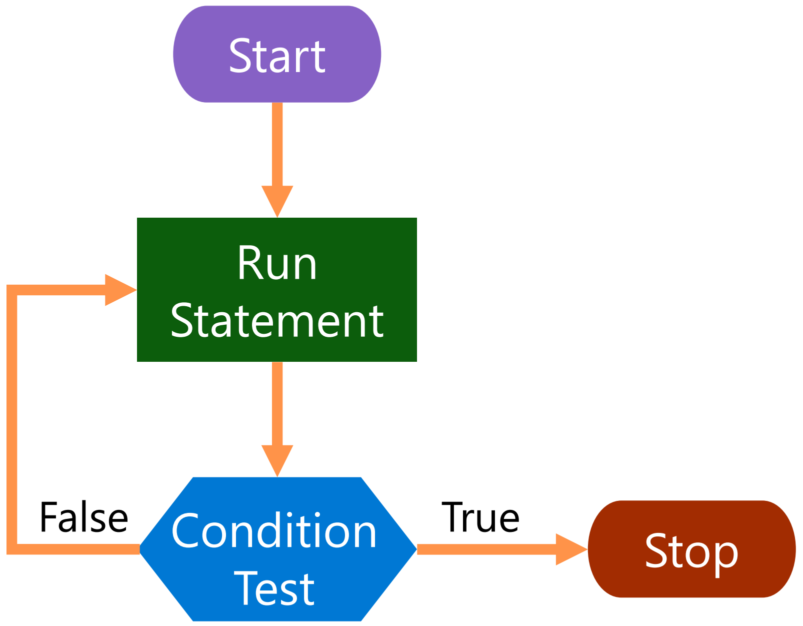
Condition:

Evaluates a comparison statement for a **$true** condition

```
Do {Statement} While (Condition)
```

```
$var = 0
Do
{
   Write-Host “user$var"
   $var++
} While ($var -lt 3)

user0
user1
user2
```

“Do Until” Loop



Description:

A Do Until loop **First** runs its statement and then tests if the condition is still **$False**

Condition:

Evaluates a comparison statement for a **$False** condition

```
Do {Statement} Until (Condition)
```

```
$var = 0
Do
{
   Write-Host "user$var"
   $var++
} Until ($var -lt 3)

user0
```

Demonstration

While Loop

Do While Loop

Do Until Loop

PowerShell Switch Command

Switch Command

Universal command to program Sequence, Selection, and Loops

Uses parameters to control behavior

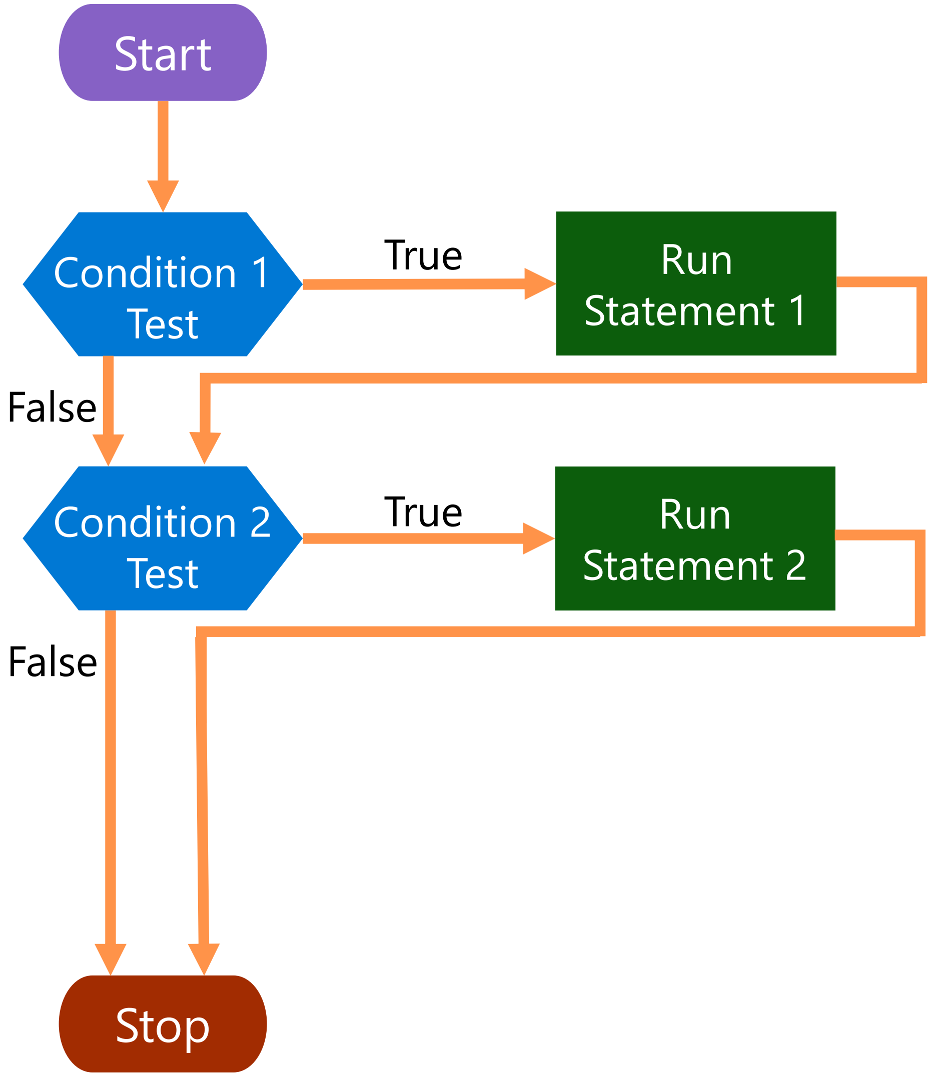
Has an option for a “Default” catch all

Can accept file paths to process contents

Called Select..Case in some other languages

Switch Command

Basic Switch



Description:

Validate **each** condition and run the statement if $True

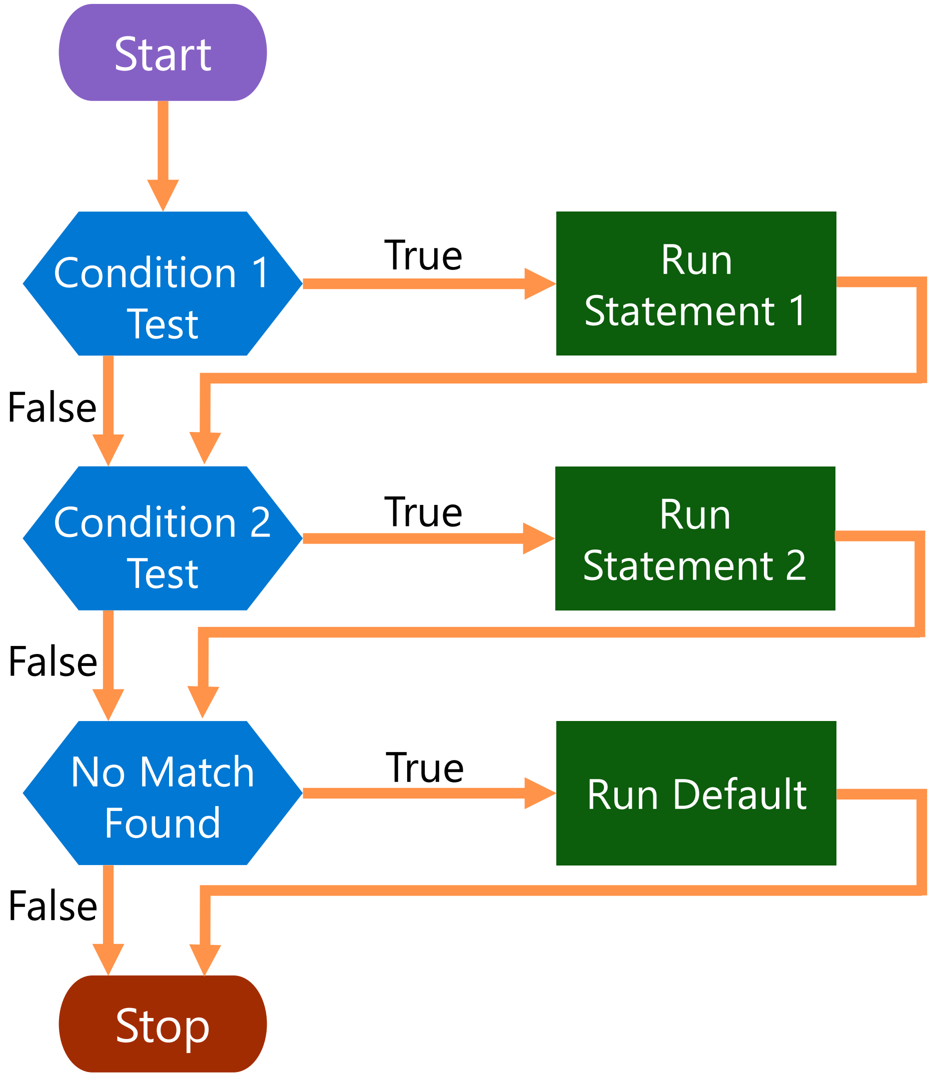
A **condition** can be a:
string, number, variable, or code block

```
Switch (<test-value>)
{
 <condition 1> {Statement 1}
 <condition 2> {Statement 2}
}
```

```
Switch ("1")
{
  "0" {Write-Host "It's 0"}
  "1" {Write-Host "It's 1"}
}

It’s 1
```

Switch Command

Switch with default case



Description:

Validate **each** condition and run statement if $True. If **no** condition matches, run the default statement

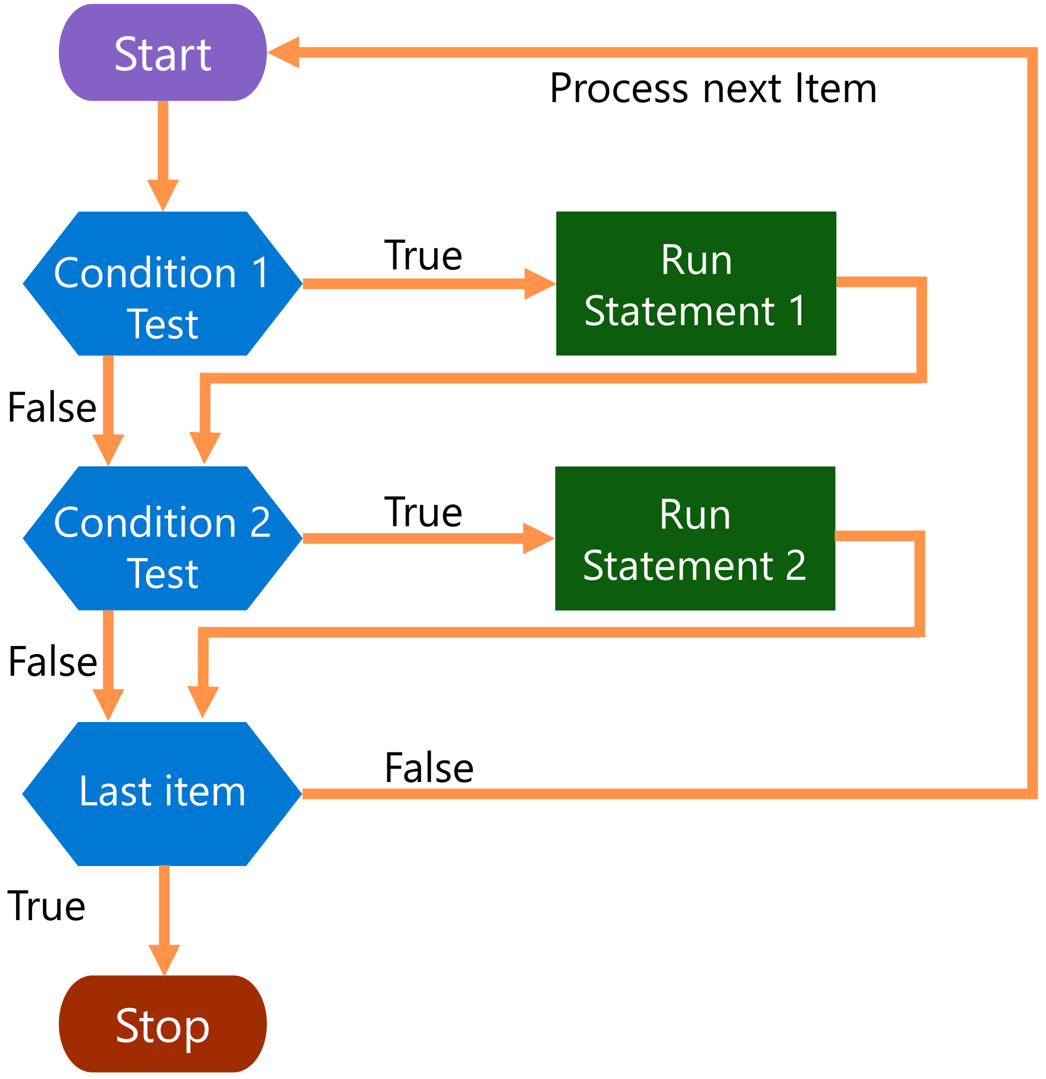
A condition can be:
string, number, variable, or code block

```
Switch (<test-value>)
{
 <condition 1> {Statement 1}
 <condition 2> {Statement 2}
 Default {Statement 3}
}
```

```
Switch ("3")
{
  "0" {Write-Host "It's 0"}
  "1" {Write-Host "It's 1"}
  Default {"It's not 0 or 1"}
}

It’s not 0 or 1
```

Switch Command

Switch with multiple values



Description:

Validate **each** condition and run statement if $True. **For each** of the items in "Test value"

A **condition** can be:
string, number, variable, or code block

```
Switch (<test-value>)
{
 <condition 1> {Statement 1}
 <condition 2> {Statement 2}
}
```

```
Switch ("1","7","0")
{
  "0" {Write-Host "It's 0"}
  "1" {Write-Host "It's 1"}
}

It's 1
It's 0
```

Demonstration

Switch

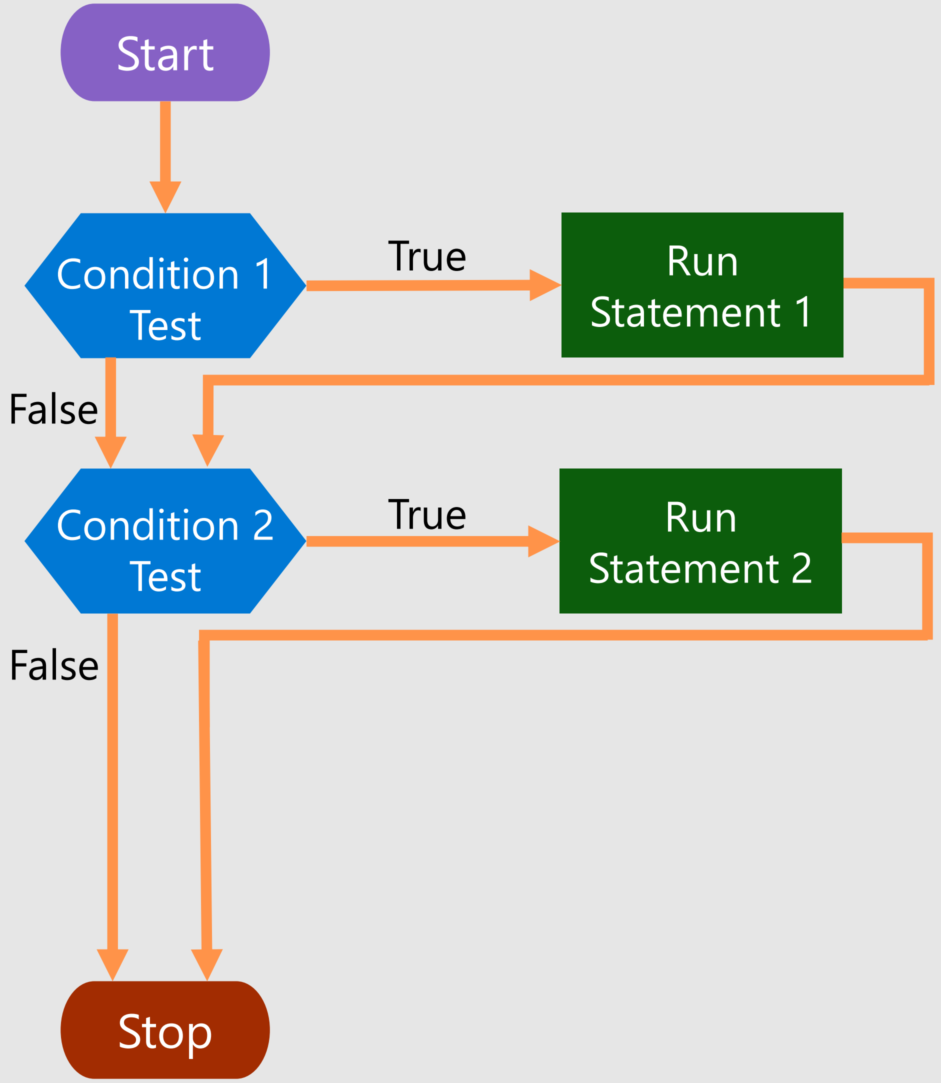
Controlling Switch Behavior With Parameters

Switch Command Parameters

Case Sensitive

```
switch -CaseSensitive ("HELLO")
{
  "hello" {"Lowercase"}
  "HELLO" {"Uppercase"}
}
```

```
Uppercase
```

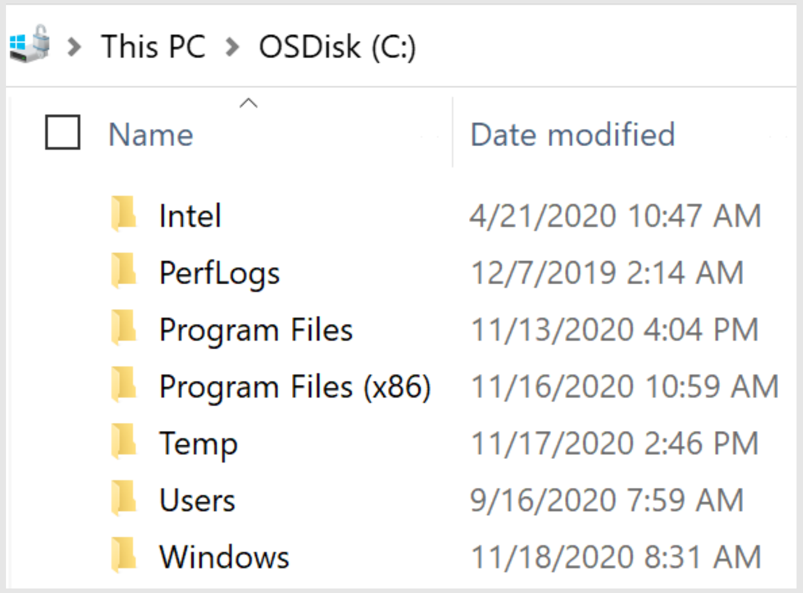


Switch Command Parameters

Case Sensitive

```
switch -Wildcard (Get-ChildItem -Path c:\)
{
   "program*" {Write-Host $_ -F Green}
   "p*s*"     {Write-Host $_ -F Yellow}
   "windows"  {Write-Host $_ -F Cyan}
}
```

```
PerfLogs
Program Files
Program Files
Program Files (x86)
Program Files (x86)
Windows
```



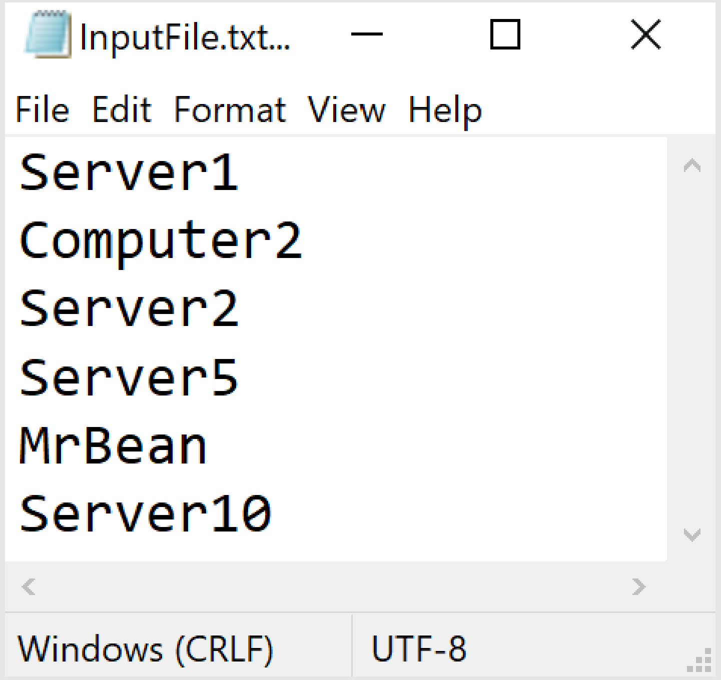
Name	Date modified
Intel	4/21/2020 10:47 AM
PerfLogs	12/7/2019 2:14 AM
Program Files	11/13/2020 4:04 PM
Program Files (x86)	11/16/2020 10:59 AM
Temp	11/17/2020 2:46 PM
Users	9/16/2020 7:59 AM
Windows	11/18/2020 8:31 AM

Switch Command Parameters

File

```
switch -File C:\Temp\InputFile.txt
{
  "Server1" {Write-Host "$_ in file" -F Gray}
  "Server2" {Write-Host "$_ in file" -F Red}
  "Server10" {Write-Host "$_ in file" -F Cyan}
}
```

```
Server1 in file
Server2 in file
Server10 in file
```



```
Server1
Computer2
Server2
Server5
MrBean
Server10
```

Switch Command Parameters

Expression Matches

```
switch (123)
{
  {$_ -lt 124}      {Write-Host $_ -ForegroundColor Green}
  {$_ -gt 200}      {Write-Host $_ -ForegroundColor Cyan}
  {$_ -match "\d*"} {Write-Host $_ -ForegroundColor Yellow}
  {$_ -like "1*"}   {Write-Host "It starts with one" -ForegroundColor Red}
}
```

```
123
123
It starts with one
```

Demonstration

Switch Parameters

PowerShell Flow Control Keywords

What Are Flow Control Keywords Used For ?

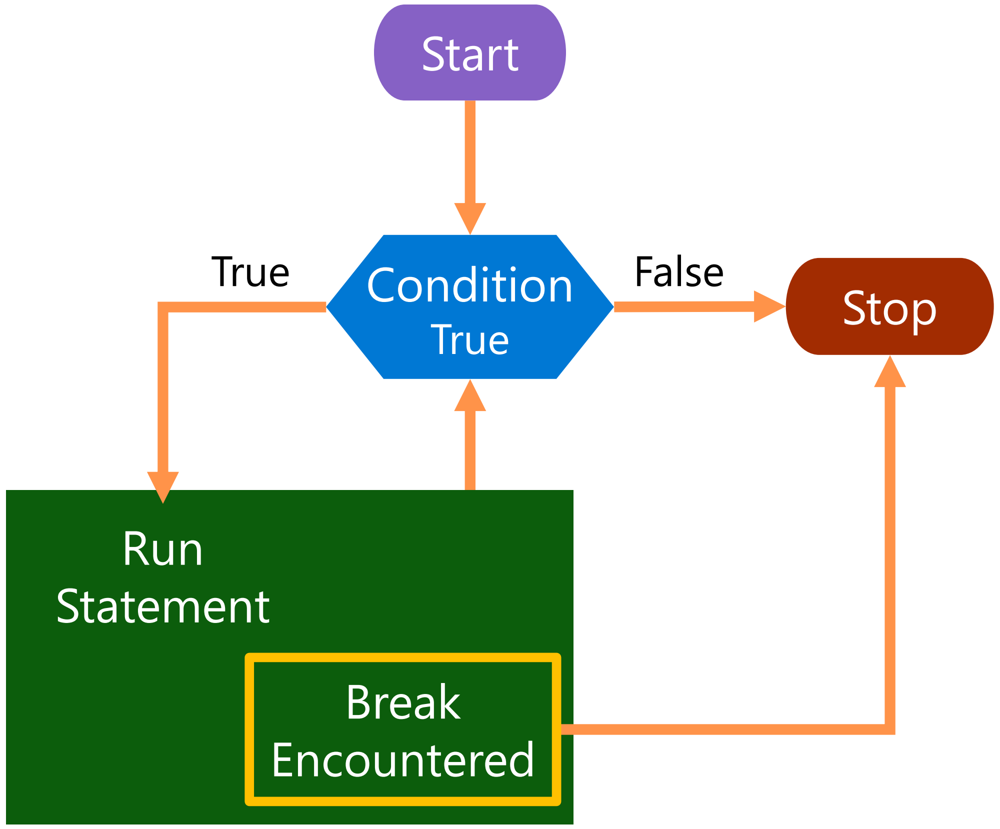
STOP LOOP PROCESSING

SKIP LOOP ITERATIONS

RETURN DATA IN A PROPER MANNER TO A CALLER

PROVIDE ERROR CONTROL TO EXTERNAL CALLER

Stopping a Loop With Break



Description:

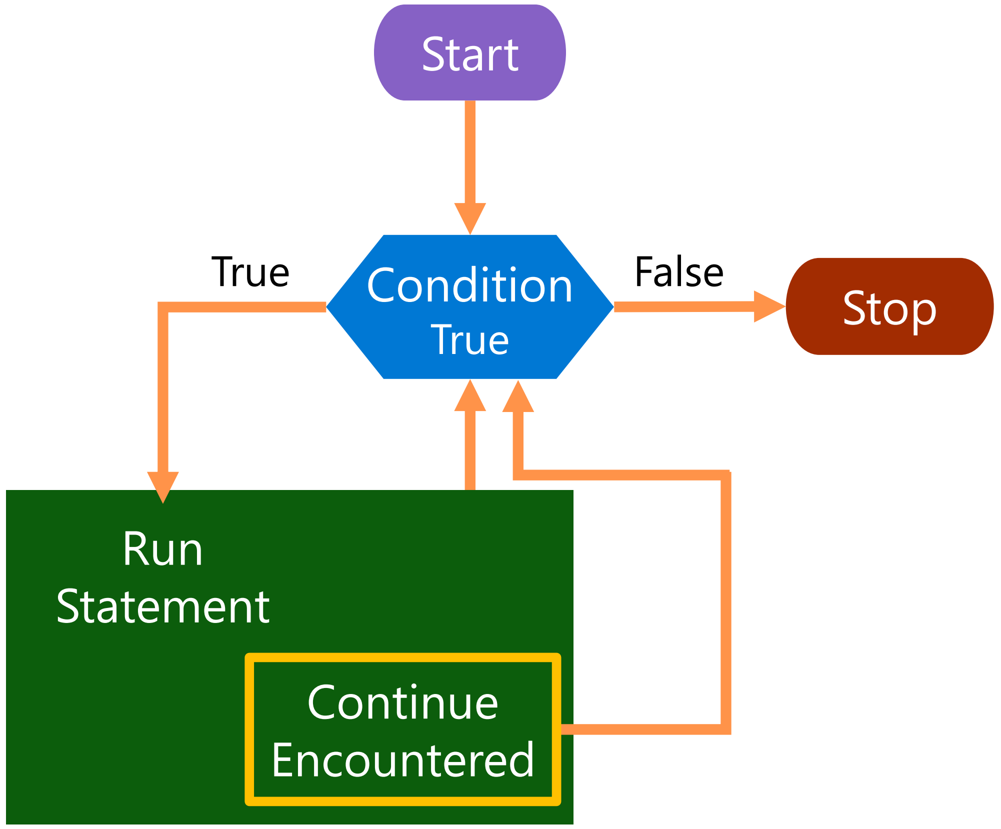
Break can be used to terminate code blocks inside the **current scope**

Use cases:

Break is suited to be used with any loop. Although it can be used in functions or scripts, it is not best practice

```
$user = 0
while ($user -lt 2)
{
   Write-Host "user$user"
   $user++
   If ($user -eq 1)
   {
      Break
   }
}

user0
```

Skipping a Loop With Continue



Description:

Continue stops the loops **current iteration**

Use cases:

Skip a **running iteration** of a loop and start with the next one.

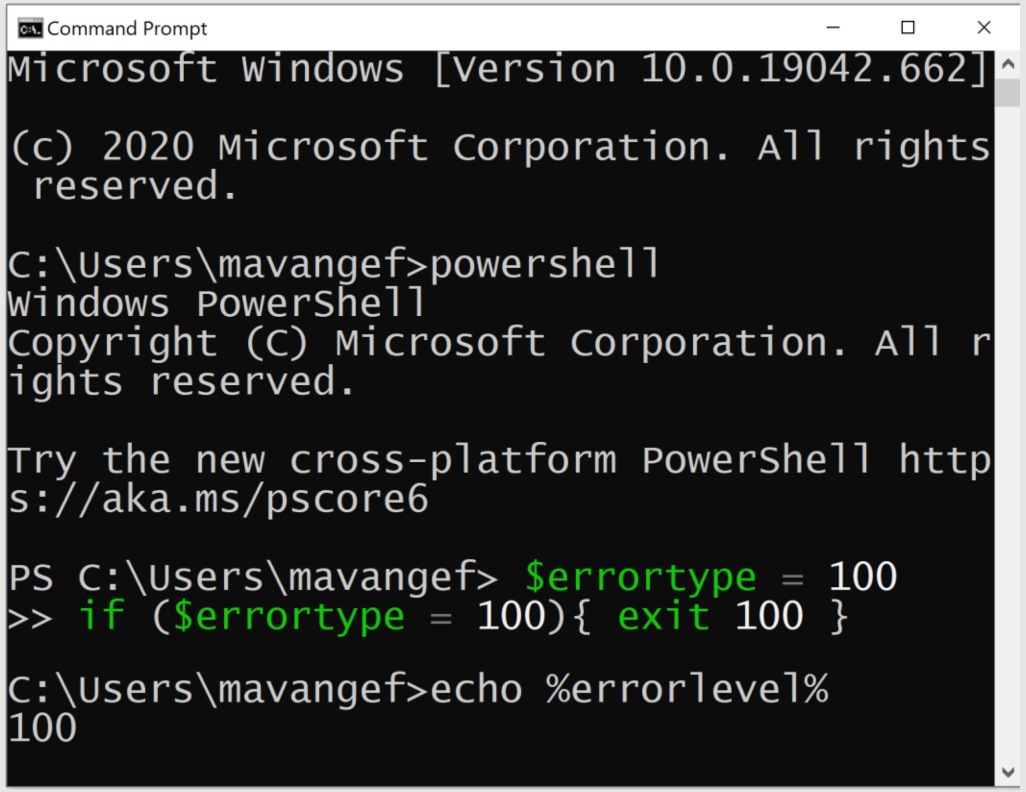
```
$user = 0
While ($user -lt 2)
{
   Write-Host "user$user"
   $user++
   If ($user -eq 1)
   {
      Continue
   }
   "processed user $user"
}
user0
user1
processed user2
```

PowerShell Return

```
Function Test-Return
{
   'Outside of return'
   return 'This value'
}

PS> $Result = Test-Return
PS> $Result

Outside of return
This value

PS> $Result.GetType().fullname

System.Object[]
```

- Used to return data to caller
- Will return every object send to pipe as a single object
- Allows explicit exit from a function
- Present for compatibility with other languages

Terminate Run space and Feedback Error Level With Exit



Command Prompt

```
Microsoft Windows [Version 10.0.19042.662]

(c) 2020 Microsoft Corporation. All rights
 reserved.

C:\Users\mavangef>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All r
ights reserved.

Try the new cross-platform PowerShell http
s://aka.ms/pscore6

PS C:\Users\mavangef> $errortype = 100
>> if ($errortype = 100){ exit 100 }

C:\Users\mavangef>echo %errorlevel%
100
```

- Only use to exit PowerShell
- Determine a numeric error schema
- Use caller tool to process error

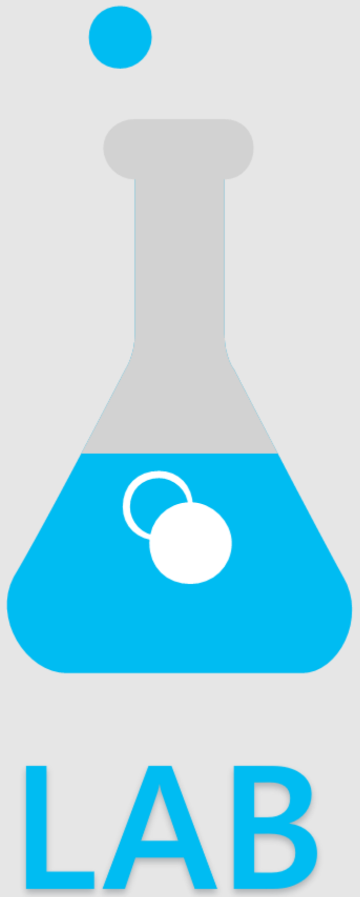
Demonstration

Flow Control Keywords

Questions?

Lab 12:
Flow Control

60 minutes



Microsoft