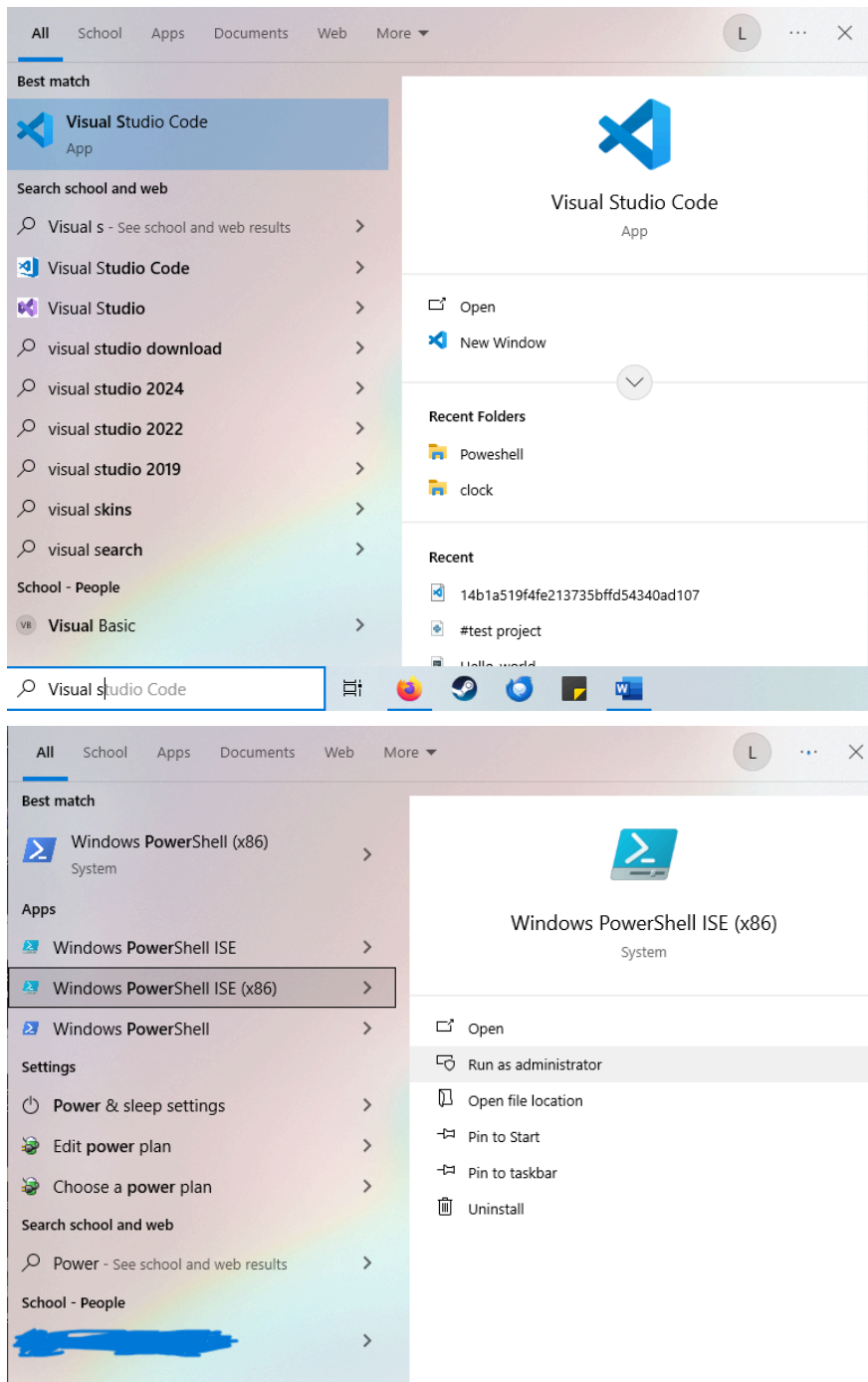


PowerShell Challenge Day 1

By Lucas Jones

Purpose - This project aims to give you 10 minutes of practice on PowerShell concepts and answer some questions about the concepts learned.

1) Open Visual Studio or PowerShell ISE as Administrator.



I prefer using Visual Studios code, so the rest of the project will use that.

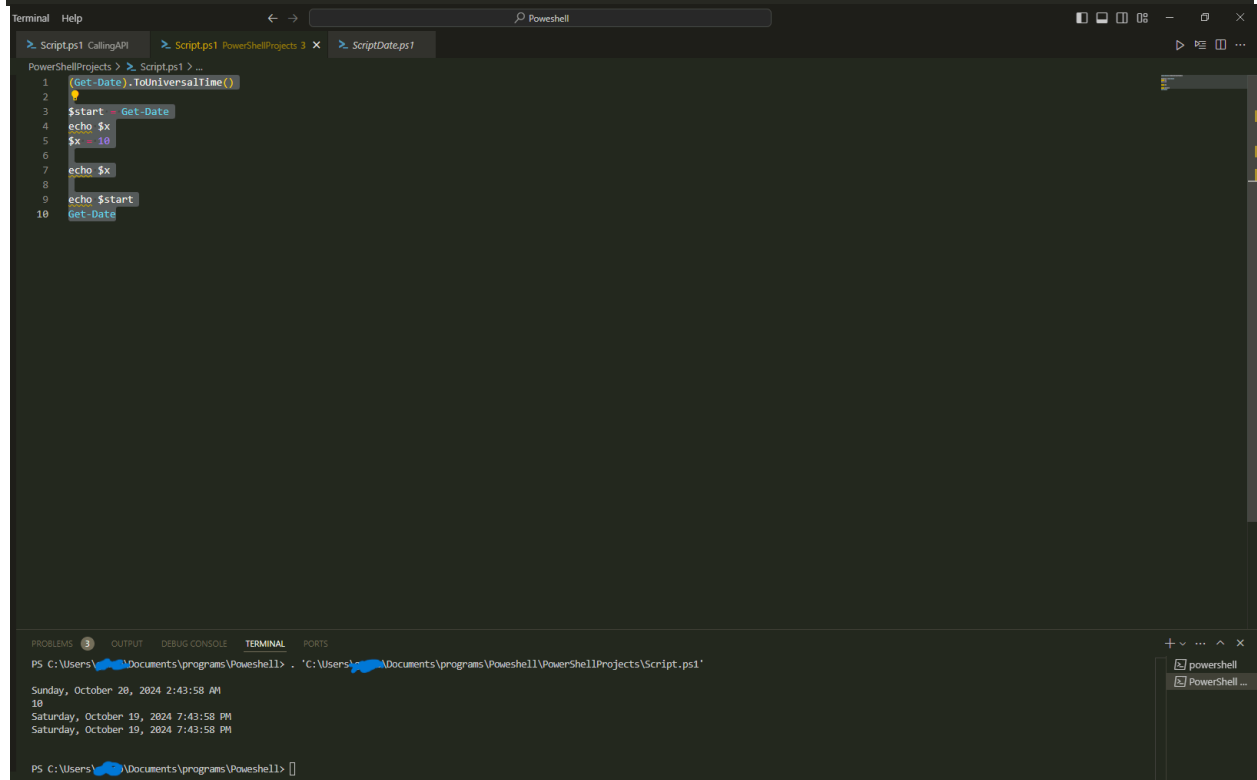
1) Practice PowerShell Syntax -

Type in the commands:

```
(Get-Date).ToUniversalTime() #Puts date in universal time

$start = Get-Date
echo $x
$x = 10
echo $x

echo $start
Get-Date
```



The code above should give you an output that looks like this.

The Get-Date Cmdlet gives you the current date of the machine being used.

The `ToUniversalTime` Cmdlet gives you the date in universal time.

Echo prints the information that is provided after the echo Cmdlet.

\$ is a variable, and you can store data or Cmdlet.

2) Some questions to ask yourself.

What does the \$ Mean before a Cmdlet?

Answer: The \$ is a variable in PowerShell, so \$start stores the Get-Date command data.

What happens when you call something with the \$, and it has nothing in it? How is this different when it has been assigned a value?

Answer: You get an error message when you call something with the \$, and nothing is assigned to it. When you assign a value to the \$ it will respond with what data you gave it, string or numerical.

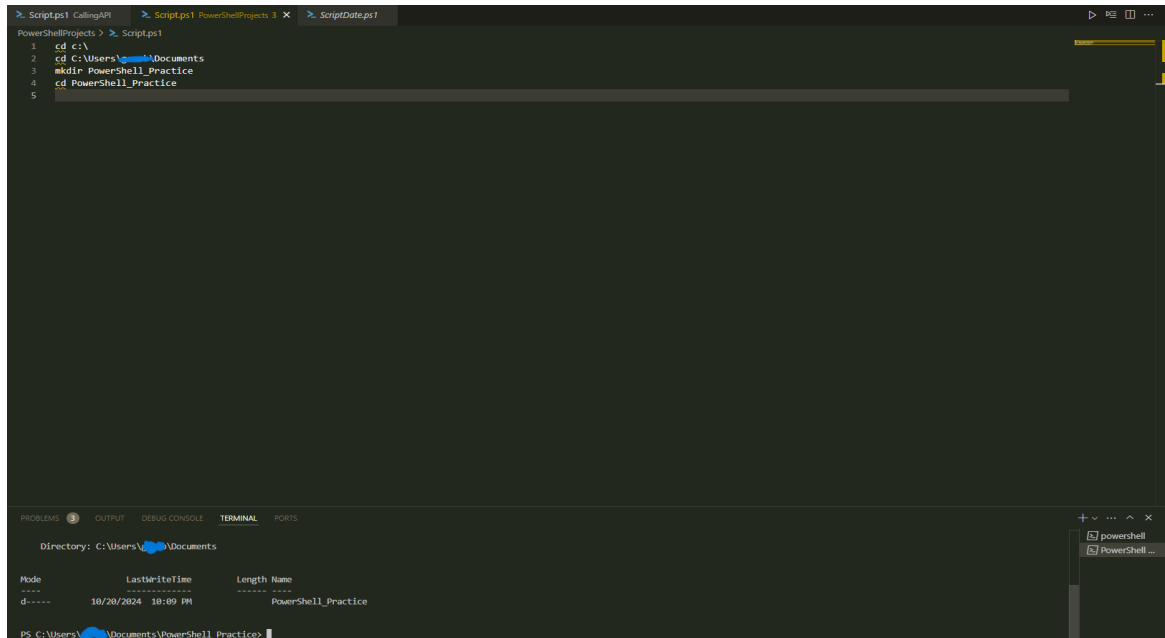
What was saved inside the \$start variable?

Answer: Get-Date Cmdlet

3) PowerShell uses basic Windows commands and a verb-Object (noun for cmdlets)

Type in these commands and see how PowerShell is very much like typical Windows commands.

```
cd c:\
cd C:\Users\<yourusername>\Documents
mkdir PowerShell_Practice
cd PowerShell_Practice
```



The screenshot shows a Visual Studio Code editor with a terminal window open. The terminal is running a PowerShell script with the following commands:

```
1 cd c:\
2 cd C:\Users\...Documents
3 mkdir PowerShell_Practice
4 cd PowerShell_Practice
5
```

The terminal output shows the directory structure:

```
Directory: C:\Users\...Documents
Mode                LastWriteTime         Length Name
----                -
d-----          10/20/2024  10:09 PM         PowerShell_Practice
```

The terminal prompt is now `PS C:\Users\...Documents\PowerShell_Practice>`.

Makes a folder in the document file

4) Some fun with testing and scripting with PowerShell

Type in the following Cmdlet

```
$start = Get-Date -UFormat "%s"
sleep 10
$end= Get-Date -UFormat "%s"
echo "This script ran for " ($end- $start) "seconds"
```

```
terminal Help
Script.ps1 CallingAPI
Script.ps1 PowerShellProjects 2 x
ScriptDate.ps1

PowerShellProjects > Script.ps1 > ...
1 $start = Get-Date -Uformat "%s"
2 sleep 10
3 $end = Get-Date -Uformat "%s"
4 echo "This script ran for " ($end - $start) "seconds"
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ FullyQualifiedErrorId : CommandNotFoundException

$
PS C:\Users\... Documents\PowerShell_Practice>
PS C:\Users\... Documents\PowerShell_Practice> . 'C:\Users\... Documents\programs\PowerShell\PowerShellProjects\Script.ps1'
This script ran for
10.0142998695374
seconds
PS C:\Users\... Documents\PowerShell_Practice>
```

So what happened with this script?

Answer: This script has two variables, the \$start and \$end. With the Cmdlet before the variables call it to sleep, this script tells you how long the downtime is that the Cmdlet went to sleep for. It prints out the seconds because of the way we used Get-Date -Uformat "%s", making the script use seconds as the time.