

Instrukcje Sterujące

Składnia instrukcji if w PowerShell

```
if (warunek1) {  
    # kod do wykonania, jeśli warunek1 jest prawdziwy  
} elseif (warunek2) {  
    # kod do wykonania, jeśli warunek2 jest prawdziwy  
} else {  
    # kod do wykonania, jeśli żaden z wcześniejszych warunków nie jest  
    prawdziwy  
}
```

Opcjonalne elementy:

- **elseif (warunek):** Dodatkowy warunek, który jest sprawdzany, jeśli poprzedni warunek (**if**) nie jest spełniony.
- **else:** Blok kodu, który jest wykonany, jeśli żaden z wcześniejszych warunków nie jest spełniony.

składnia instrukcji for w PowerShell jest następująca:

```
for (inicjalizacja; warunek; iteracja)  
{  
    # Blok kodu do wykonania  
}
```

- **Inicjalizacja:** Jest to wyrażenie, które jest wykonane tylko raz na początku pętli.
- **Warunek:** Jest to wyrażenie logiczne, które jest sprawdzane przed każdą iteracją. Jeśli warunek jest prawdziwy, blok kodu zostanie wykonany.
- **Iteracja:** Jest to wyrażenie, które jest wykonane po każdym przejściu przez pętlę.

Instrukcje **break** i **continue** w PowerShell są używane do kontrolowania przepływu w pętlach. Oto ich zastosowanie:

break

Instrukcja **break** służy do natychmiastowego zakończenia pętli, bez względu na to, czy warunek zakończenia pętli został spełniony. Po jej wykonaniu, program przechodzi do pierwszej instrukcji po pętli.

continue

Instrukcja **continue** służy do pominięcia reszty kodu w bieżącym cyklu pętli i natychmiastowego przejścia do następnego cyklu (po ponownym sprawdzeniu warunku pętli).

Pętla **foreach**

w PowerShell służy do iterowania przez elementy kolekcji (np. tablicy, listy, słownika itd.) i wykonania pewnych działań dla każdego z tych elementów. Poniżej znajduje się jej podstawowa składnia:

```
foreach ($element in $kolekcja) {  
    # Kod do wykonania dla każdego elementu  
}
```

Gdzie:

- **\$element** to zmienna, która przyjmuje wartość każdego elementu z kolekcji **\$kolekcja**.
- **\$kolekcja** to zbiór elementów, przez które pętla będzie iterować.

Do-While

W przypadku pętli **do-while**, kod w bloku **do** zostanie wykonany co najmniej raz, a następnie zostanie sprawdzony warunek w **while**. Jeśli warunek jest prawdziwy, kod w bloku **do** zostanie wykonany ponownie.

```
$licznik = 0  
do {  
    Write-Host "Licznik wynosi: $licznik"  
    $licznik++  
} while ($licznik -lt 5)
```

Do-Until

W przypadku pętli **do-until**, kod w bloku **do** zostanie również wykonany co najmniej raz. Pętla będzie kontynuowana, dopóki warunek w **until** nie będzie prawdziwy.

```
$licznik = 0  
do {  
    Write-Host "Licznik wynosi: $licznik"  
    $licznik++  
} until ($licznik -eq 5)
```

Do

Instrukcja **do** sama w sobie nie robi nic specjalnego. Jest to jednak blok kodu, który można wykorzystać w połączeniu z **while** lub **until**.

```
$licznik = 0  
  
do {  
    Write-Host "To się wykona tylko raz."  
    $licznik++  
}
```