

# Podstawowe wytyczne pisania kodu w PowerShell (Style Guide)

## 1. Cele stylu

Dobre praktyki pisania kodu w PowerShell zwiększą czytelność, ułatwiają utrzymanie i minimalizują ryzyko błędów. Poniższy dokument zawiera podstawowe wytyczne zgodne z rekomendacjami zespołu PowerShell Team oraz modułu PSScriptAnalyzer.

## 2. Konwencje nazewnictwa

### 2.1 Funkcje

- Funkcje powinny być nazywane w formacie Verb-Noun (czasownik-rzeczownik), np.:
  - `Get-UserData`
  - `Set-ServerConfig`
  - `New-Report`
- Używaj tylko zatwierdzonych czasowników (Approved Verbs).

Sprawdzenie dostępnych czasowników:

#### *Get-Verb*

---

### 2.2 Zmienne

- Stosuj styl camelCase, np.:  
`$userName`  
`$logPath`
- Nazwy powinny być opisowe, ale zwięzłe.

### 2.3 Stałe i wartości niezmienne

- Jeśli używasz stałych, zapisuj je wielkimi literami:  
`$MAX_COUNT = 10`

## 3. Formatowanie kodu

### 3.1 Bloki logiki

- Nawiąsy klamrowe otwieraj w tej samej linii co instrukcja sterująca:

```
if ($value -gt 10) {  
    Write-Host "OK"  
}
```

---

### 3.2 Pipeline

- Długie potoki pipeline dziel na kolejne linie:

```
Get-Process |  
Where-Object { $_.CPU -gt 100 } |  
Sort-Object CPU -Descending
```

---

### 3.3 Spacje i odstępy

- Jedna spacja po przecinku.
- Jedna spacja po operatorach logicznych i porównania.
- Unikaj zbędnych spacji na końcu linii.

## 4. Unikanie aliasów i skrótów

Aliasów (np. ls, cat, gc) nie stosuj w skryptach produkcyjnych ani edukacyjnych.

Zamiast tego używaj pełnych cmdletów:

- *Zamiast ls → Get-ChildItem*
- *Zamiast cat → Get-Content*
- *Zamiast gm → Get-Member*

## 5. Parametry i deklaracja funkcji

- Parametry funkcji deklaruj w bloku param() na początku:

```
function Get-Data {  
    param(  
        [string]$Name,  
        [int]$Age  
    )  
}
```

---

## 6. Komentarze

- Komentarz blokowy dla opisów funkcji:

```
<#  
Opis działania funkcji  
Autor, data  
#>
```

---

- Komentarze jedno-liniowe używaj oszczędnie, ale tam gdzie to potrzebne:  
`# Sprawdzenie poprawności danych`

## 7. Używanie PSScriptAnalyzer

PSScriptAnalyzer jest narzędziem analizującym jakość kodu (odpowiednik linters z Python).

Instalacja modułu:

```
Install-Module PSScriptAnalyzer
```

---

Analiza skryptu:

```
Invoke-ScriptAnalyzer -Path .\skrypt.ps1
```

---

## 8. Linki i źródła oficjalne

- PowerShell Style Guide (MS Docs):

<https://learn.microsoft.com/powershell/scripting/learn/deep-dives/effective-powershell>

---

- PSScriptAnalyzer (GitHub):

<https://github.com/PowerShell/PSScriptAnalyzer>

---

- Approved Verbs (MS Docs):

<https://learn.microsoft.com/powershell/scripting/developer/cmdlet/approved-verbs-for-windows-powershell-commands>

---

## 9. Podsumowanie

PowerShell nie posiada jednego odpowiednika PEP-8 jak Python, jednak istnieje spójny zestaw zasad style guide, wsparty przez narzędzie PSScriptAnalyzer. Przestrzeganie zasad zwiększa jakość, czytelność i skalowalność skryptów.