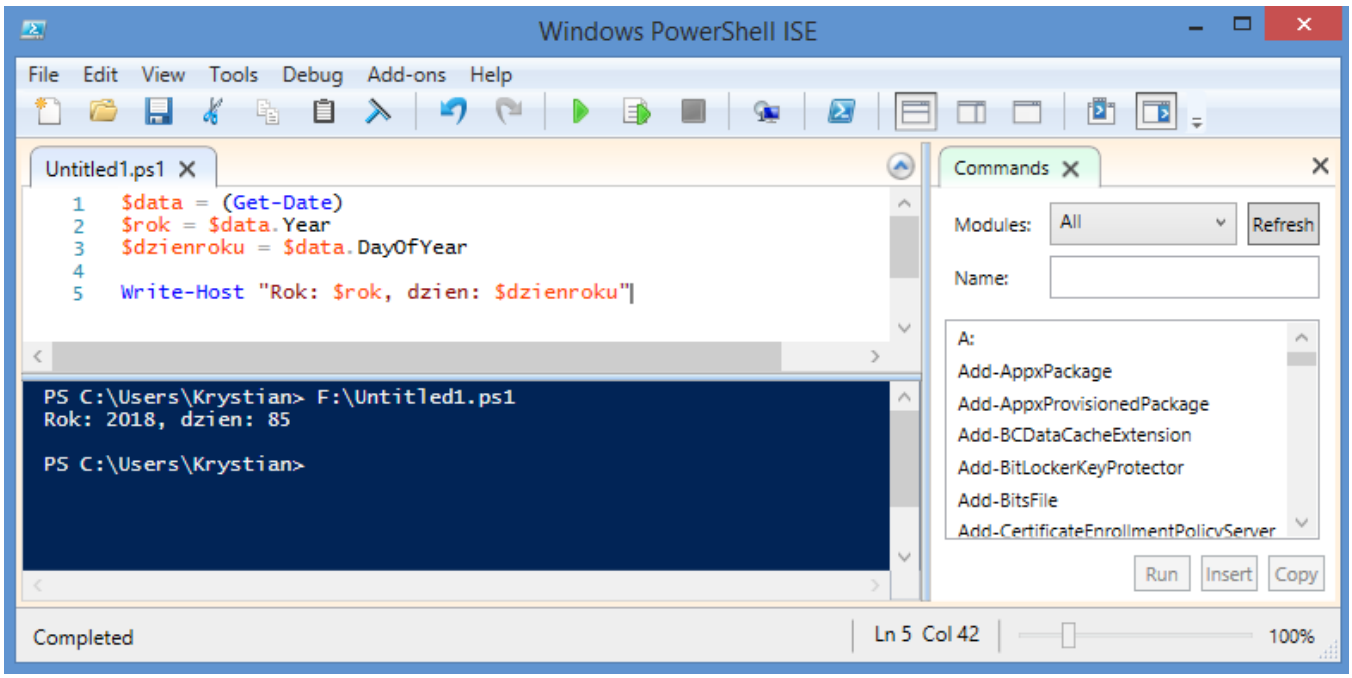
PowerShell – Laboratorium I oraz II

1. Wstęp

PowerShell to interpreter poleceń bardziej rozbudowany w stosunku do wcześniejszych interpreterów COMMAND.COM. Wprowadza on możliwość wykorzystania obiektów, właściwości i jest zintegrowany z .NET Framework. Począwszy od Windows XP SP2 jest on wbudowany w system Windows, a zatem jest interpreterem ogólnodostępnym (w niektórych wersjach systemu Windows jest komponentem opcjonalnym). Najnowszą wersją jest 6.0.

Uruchomienie interpretera:

Ctrl + R → powershell (tryb podstawowy)
Ctrl + R → **powershell ISE** (tryb graficzny z edytorem skryptów)



Rysunek 1. Tryb graficzny z edytorem skryptów (PowerShell ISE)

2. Polecenia

PowerShell pozwala na wykonywanie poleceń wykorzystywanych przez interpreter COMMAND.COM (np. dir, tasklist), jednak dopiero dedykowane polecenia pozwalają na pełne wykorzystanie jego potencjału, przyjmują one następującą postać:

Czasownik-Rzeczownik

Przykładowe polecenia wyglądają zatem następująco:

Write-Host tekst	- wypisanie tekstu na ekran
Get-Date	- pobranie aktualnej daty
Get-Content „plik.txt”	- wyświetlenie zawartości pliku
Get-Help polecenie	- wyświetlenie pomocy

3. Zmienne

Zmienne deklaruje się dopisując przedrostek $:

$zm1 = 15

Opcjonalnie możliwe jest podanie typu zmiennej:

[string] $zm2 = "tekst"

Pobranie wartości zmiennej od użytkownika umożliwia polecenie Read-Host:

$zm3 = Read-Host "Podaj liczbe"

Operacje arytmetyczne na zmiennych wykonuje się w sposób standardowy:

$zm4 = $zm1 + 15

Zmienne można także tworzyć w bardziej rozbudowany sposób, wykorzystując polecenie PowerShell-a:

New-Variable -Name zm5 -Description "opis" -Value "wartosc"

Zadanie 1.

Wczytać od użytkownika dwie liczby (bez określenia typu zmiennych). Przypisać wynik mnożenia tych zmiennych do trzeciej zmiennej i wyświetlić wynik. Wykonać to samo podając typ zmiennych: **int**

4. Operacje na plikach i katalogach

Get-ChildItem	- pobranie listy plików i folderów
New-Item "nazwa" –ItemType "typ"	- utworzenie pliku lub folderu (typ = "directory" lub "file")
$plik = Get-Item	- wczytanie **informacji** o pliku do zmiennej
Copy-Item	- kopiowanie plików lub folderów
Move-Item	- przenoszenie plików lub folderów
Remove-Item	- usuwanie plików lub folderów

Przykład - utworzenie pliku, wyświetlenie informacji o pliku i nadanie mu atrybutu:

```
New-Item "plik.txt" -ItemType "File" -Force
$plik = Get-Item "plik.txt"
Write-Host Atrybuty: $plik.Attributes
Write-Host Data utworzenia: $plik.CreationTime
Write-Host Rozmiar pliku: $plik.Length
$plik.Attributes = "System"
Write-Host Zmieniony atrybuty: $plik.Attributes
```

Zadanie 2.

Wyświetlić pomoc odnośnie komend z podpunktu 4, a także pomoc dla polecenia Get-Help, następnie:

- Utworzyć katalog *Dom*
- Utworzyć katalog *Dom/Drzwi* oraz *Dom/Okno*
- Utworzyć plik *Dom/Drzwi/klamka.txt*
- Przekopiować plik *klamka.txt* do katalog *Okno* zmieniając jego nazwę na *okienna.txt*
- Ustawić atrybut pliku *klamka.txt* na ukryty, a *okienna.txt* na tylko do odczytu (**lista atrybutów**: ReadOnly, Hidden, System, Archive, Normal, Temporary, Compressed, Offline, Encrypted)
- Wyświetlić zawartość pliku *okienna.txt*
- Wyświetlić listę plików i folderów z katalogu *Dom* (zawartość wszystkich katalogów -Recurse)
- Znajdź sposób na wyświetlenie plików ukrytych

Rozbudowany opis komend i PowerShella można znaleźć pod następującym adresem:

http://www.iisi.pcz.pl/index.php/pl/do-pobrania?func=startdown&id=926

5. Instrukcje warunkowe i pętle

Instrukcje warunkowe przyjmują następującą postać:

```
If (warunek) {
  blok instrukcji
} elseif (warunek) {
  blok instrukcji
} else {
  blok instrukcji 3
}
```

Uwaga: ponieważ znaki > itp. są **zarezerwowane dla potoków**, do porównywania wartości używane są następujące operatory (dotyczy to także pętli):

operator	**opis**	**przykład**
-eq	operator równości (**eq**ual)	5 –eq $val
-ne	operator nierówności (**n**ot **e**qual)	5 –ne $val
-ge	większe lub równe (**g**reater or **e**qual)	5 –ge $val
-gt	większe (**g**reater **t**han)	5 –gt $val
-le	mniejsze lub równe (**l**ower or **e**qual)	5 –le $val
-lt	mniejsze (**l**ower **t**han)	5 –lt $val
-like	odpowiadające wzorcowi	"test" –like "t*t"
-notlike	nie odpowiadające wzorcowi	"test" –notlike "t*t"
-contains	operator zawiera	"test" –contains "te"
-notcontains	operator nie zawiera	"test" –notcontains "te"
-is	porównanie typu zmiennej	$val -is [int]
-isnot	negacja porównania typu ziennej	$val –isnot [int]
-and	iloczyn logiczny	(warunek) –and (warunek)
-or	suma logiczna	(warunek) –or (warunek)
-xor	alternatywa wykluczająca	(warunek) –xor (warunek)
-not	negacja	-not $val

Przykład wykorzystania instrukcji warunkowych:

```
[int]$a = Read-Host "Podaj A"
[int]$b = Read-Host "Podaj B"

if ($a -lt 10 -and $b -lt 10) {
   Write-Host A i B sa mniejsze niz 10
} elseif ($a -eq 15) {
   Write-Host A jest rowne 15
}
```

Instrukcje switch przyjmują następującą postać:

```
switch (zmienna)
{
   wartość { blok instrukcji }
   wartość { blok instrukcji }
   wartość { blok instrukcji }
   default { blok instrukcji }
}
```

Pętle przyjmują następującą postać:

```
for (deklaracje; warunek; operacje) {
        instrukcje
}
```

```
foreach (element in kolekcja) {
        instrukcje
}
```

```
while (warunek) {
        instrukcje
}
```

```
do {
        instrukcje
while (warunek)
```

Przykład wykorzystania pętli (należy zwrócić uwagę na wykorzystanie **odpowiednich operatorów**):

```
for ($i = 0; $i -lt 10; $i++) {
   Write-Host $i
}
```

Zadanie 3*. (rozwiązanie zadań oznaczonych * należy umieścić w sprawozdaniu lub pokazać podczas zajęć).

- Przypisz do zmiennej $lista wynik polecenia Get-ChildItem –Force
- Wyświetl zawartość zmiennej $lista
- Za pomocą pętli **foreach** wyświetl kolejno elementy kolekcji lista - foreach($element in $lista) { …
- Sprawdź jakie właściwości dodatkowe ma każdy element
  (w tym celu należy postawić znak kropki po $element wewnątrz pętli)
- Odnajdź rozmiar, atrybuty i nazwę elementu, wyświetl te informacje, zwróć uwagę na rozmiar katalogu
- Utwórz zmienną która pozwoli zsumować rozmiar wszystkich elementów i wyświetl obliczony rozmiar
- Wewnątrz pętli dodaj warunek który nie pozwala zliczać rozmiaru katalogów

6. Potoki i strumienie

PowerShell, tak jak i COMMAND.COM pozwala na wykorzystywanie strumieni i potoków.

polecenie > plik	- zapis wyniku polecenia do pliku
polecenie >> plik	- dopisanie wyniku polecenia do pliku
polecenie 2> plik	- przekierowanie komunikatu błędu do pliku
polecenie 3> plik	- przekierowanie komunikatu ostrzeżenia do pliku
polecenie 4> plik	- przekierowanie komunikatu uzupełnienia do pliku
polecenie 5> plik	- przekierowanie komunikatu debugowania do pliku
polecenie1 | polecenie2	- przekazanie wyniku polecenia1 do polecenia2

Zadanie 4.

- Przekazać wynik polecenia Get-ChildItem do polecenia Sort-Object
- Polecenie Sort-Object może posortować wyniki według dowolnej kolumny, należy wywołać je w taki sposób, aby sortowało wyniki według kolumny Mode
- Przekazać wynik powyższych poleceń do polecenia Out-GridView

7. Przetwarzanie tekstu

PowerShell udostępnia nie tylko dostęp do szeregu komponentów środowiska .NET, ale także posiada analogicznie zbudowane metody do operacji na tekście i tablicach, m.in.:

$tekst.Length	- zwraca długość tekstu
$tekst.Contains($ciag)	- zwraca prawdę gdy tekst zawiera dany ciąg
$tekst.IndexOf($ciag)	- zwraca indeks wystąpienia szukanego ciągu
$tekst.IndexOf($ciag, $szukajod)	- zwraca indeks wystąpienia szukanego ciągu
$tekst.Insert($gdzie, $ciag)	- wstawia tekst do zmiennej
$tekst.Remove($start, $ile)	- usuwa tekst poczynając od indeksu $gdzie
$tekst.Replace($stary, $nowy)	- zamienia tekst na nowy ciąg
$tablica = $tekst.Split($podzial)	- tworzy z tekstu tablicę dzieląc według $podzial
$tekst = [string]::Join($separator, $tablica)	- łączy tablicę w jednolity tekst
$tekst.Substring($od, $ile)	- pobiera z tekstu określony wycinek

Zadanie 5*. (rozwiązanie zadań oznaczonych * należy umieścić w sprawozdaniu lub pokazać podczas zajęć).

- Przypisać do zmiennej [string]$miesiace nazwy miesięcy: (Get-Culture).DateTimeFormat.MonthNames
- Wyświetlić zawartość zmiennej $miesiace
- Ze zmiennej miesiące usunąć miesiąc Luty
- Rozbić zmienną miesiące na tablicę
- Za pomocą pętli for przejść po elementach tablicy dopisując do każdego miesiąca liczbę jego dni
  Pobranie liczby dni danego miesiąca:
  $indeks = (Get-Culture).DateTimeFormat.MonthNames.IndexOf($tablica[$i]) + 1
  $dni = (Get-Culture).Calendar.GetDaysInMonth(2018, $indeks)
- Po zakończeniu działania pętli złączyć elementy tablicy w nowy ciąg tekstowy
- Wyświetlić zawartość nowego ciągu tekstowego

8. Zadania dodatkowe

Zadanie 6.

- Przeanalizować działanie następującej komendy:
  Get-Process | Where-Object { $_.Name -like "c*" }
- Zmodyfikować ją tak, aby wyświetlane były tylko procesy zajmujące więcej niż 50MB pamięci RAM

Zadanie 7.

- Pobrać od użytkownika dwie wartości liczbowe oraz typ działania (dodawanie, odejmowanie, mnożenie, dzielenie)
- Wykorzystując instrukcję **switch** wykonać odpowiednie operacje matematyczne
- Wyświetlić wynik

Zadanie 8.

- Napisz skrypt tworzący pliki o losowych nazwach (wykorzystaj Get-Random oraz pętle for)

Zadanie 9.

- Zapoznaj się z rozdziałami 2.2, 2.9, 2.10, 2.11, 2.13, 2.14, 3.1, 3.3 instrukcji
  http://www.iisi.pcz.pl/index.php/pl/do-pobrania?func=startdown&id=926
- Zaproponuj własne skrypty