Typy zmiennych – info

Deklarowanie zmiennych określonego typu

Korzyści płynące z wykorzystania typów

Metody daty i czasu

Metody typu napisowego string

Sprawdzenie jakiego typu jest zmienna

Konwersje typów

Wiemy, że ze zmiennymi można pracować nie określając typów, ale jeżeli chcesz określić typ można to zrobić

Przed nazwą zmiennej w [] wpisać nazwę typu np. [int]$number=100

```
22  [int]$number=100
```

```
PS C:\WINDOWS\system32> [int]$number=100

PS C:\WINDOWS\system32>
```

Od teraz mam zmienną, która jest liczbą

```
23  $number
```

```
PS C:\WINDOWS\system32> $number
100
```

Próba wpisania do zmiennej jakiegoś tekstu

```
24  $number="tekst"
```

```
PS C:\WINDOWS\system32> $number="tekst"
Cannot convert value "tekst" to type "System.Int32". Error: "Nieprawidłowy format ciągu wejściowego."
At line:1 char:1
+ $number="tekst"
+ ~~~~~~~~~~~~~~~
    + CategoryInfo          : MetadataError: (:) [], ArgumentTransformationMetadataException
    + FullyQualifiedErrorId : RuntimeException
```

Zmiennej tekstowej nie udało się skonwertować do liczby.

```
25  [datetime]$date="2031-09-28"
26  $date
```

```
PS C:\WINDOWS\system32> [datetime]$date="2031-09-28"

PS C:\WINDOWS\system32> $date

niedziela, 28 września 2031 00:00:00
```

PowerShell wyświetla datę inaczej

Po co są typy czy praca bez określania typów nie jest wygodniejsza, jeśli zmienna posiada przypisany typ to można korzystać z całego zakresu metod i właściwości takiego typu można np. dodać do daty określoną liczbę dni, sprawdzić dzień tygodnia miesiąc, konwertować datę do tekstu w określonym formacie, sprawdzić w jaki dzień wypadnie 20 dni po 28 września

```
27  $date.AddDays(20)
```

```
PS C:\WINDOWS\system32> $date.AddDays(20)

sobota, 18 października 2031 00:00:00
```

A jaki dzień to 18 październik?

```
29  $date.AddDays(20).DayOfWeek
```

```
PS C:\WINDOWS\system32> $date.AddDays(20).DayOfWeek
Saturday
```

Sobota

A jaka data będzie 13 dni przed 28 września 2031

```
30  $date.AddDays(-13)
```

```
PS C:\WINDOWS\system32> $date.AddDays(-13)

poniedziałek, 15 września 2031 00:00:00
```

Podobnie można pracować ze zmiennymi typu logicznego

```
31  [Boolean]$bool=$true
32  $bool
```

```
PS C:\WINDOWS\system32> [Boolean]$bool=$true

PS C:\WINDOWS\system32> $bool
True
```

Jeżeli przypiszę do zmiennej bool tekst "False" to będzie błąd

```
33  $bool="False"
```

```
PS C:\WINDOWS\system32> $bool="False"
Cannot convert value "System.String" to type "System.Boolean". Boolean parameters accept only Boolean va
lues and numbers, such as $True, $False, 1 or 0.
At line:1 char:1
+ $bool="False"
+ ~~~~~~~~~~~~
    + CategoryInfo          : MetadataError: (:) [], ArgumentTransformationMetadataException
    + FullyQualifiedErrorId : RuntimeException
```

Komunikat o błędzie informuje co może być wstawione jako wartość do zmiennej logicznej

Masz predefiniowaną wartość $True $False lub 1 i 0

Można wstawiać też różne wyrażenia, które zwracają wartość prawda lub fałsz np. sprawdzenie czy 1 > 0

```
PS C:\WINDOWS\system32> $bool=0

PS C:\WINDOWS\system32> $bool
False
```

Aby sprawdzić jakie metody ma określony typ wystarczy zmienna określonego typu potokiem przesłać do get-member tam zobaczysz nazwy wszystkich metod, które mogą być wykorzystywane w pracy z tym typem.

Zmienne napisowe

```
36  [string]$s="Hello"
37  $s
```

```
PS C:\WINDOWS\system32> [string]$s="Hello"

PS C:\WINDOWS\system32> $s
Hello
```

Charakterystyczny jest tu zestaw metod

```
38  $s.Contains("He")
```

```
PS C:\WINDOWS\system32> $s.Contains("He")
True
```

Sprawdzamy czy w napisie ”Hello” jest ciąg znaków ”He”

```
39  $S.IndexOf("l")
```

```
PS C:\WINDOWS\system32> $S.IndexOf("l")
2
```

Zdaniem PowerShell występuje na drugiej pozycji napisu ”Hello” 0 PowerShell liczy od 0

Założymy, że w zmiennej tekstowej mamy napis porozdzielany przecinkami, wiemy, że nie jest to tablica napisu, ale można z tego zrobić tablice

```
40  $s="jeden,dwa,trzy,cztery,pięć"
41  $s.Split(",")
```

```
PS C:\WINDOWS\system32> $s="jeden,dwa,trzy,cztery,pięć"

PS C:\WINDOWS\system32> $s.Split(",")
jeden
dwa
trzy
cztery
pięć
```

Metoda Split odczytuje co jest znakiem rozdzielającym, w nawiasie wskazujemy na przecinek.

Sprawdzimy, czy zmienna jest typu string (napis) użyjemy funkcji is

```
42  $s -is [string]
```

```
PS C:\WINDOWS\system32> $s -is [string]
True
```

Sprawdzimy, czy zmienna jest typu int (liczba) użyjemy funkcji is

```
43  $s -is [int]
```

```
PS C:\WINDOWS\system32> $s -is [int]
False
```

Konwersje, które wykonuje PowerShell podczas pracy z zmiennymi

```
44  $s=10
45  $s
```

```
PS C:\WINDOWS\system32> $s=10

PS C:\WINDOWS\system32> $s
10
```

```
42  $s -is [string]
43  $s -is [int]
```

```
PS C:\WINDOWS\system32> $s -is [string]
True

PS C:\WINDOWS\system32> $s -is [int]
False
```

- Jak widzimy liczba została skonwertowana do napisu i to napis jest zawartością zmiennej.

Czy da się do zmiennej typu int przypisać string

```
44    $number=$s
45    $number
```

```
PS C:\WINDOWS\system32> $number=$s

PS C:\WINDOWS\system32> $number
10
```

Ile to jest 10 + 10, gdy jedna zmienna jest liczbą a druga napisem

```
46    $number+$s
```

```
PS C:\WINDOWS\system32> $number+$s
20
```

Jeżeli w zmiennej napisowej znajduje się coś czego nie można przekonwertować na liczbę to

```
47    $s="ten"
```

```
PS C:\WINDOWS\system32> $s="ten"

PS C:\WINDOWS\system32> $number=$s
Cannot convert value "ten" to type "System.Int32". Error: "Nieprawidłowy format ciągu wejściowego."
At line:1 char:1
+ $number=$s
+ ~~~~~~~~~~
    + CategoryInfo          : MetadataError: (:) [], ArgumentTransformationMetadataException
    + FullyQualifiedErrorId : RuntimeException
```

będzie błąd

Deklarowanie zmiennych określonego typu

Korzyści płynące z wykorzystania typów

Metody daty i czasu

Metody typu napisowego string

Sprawdzenie jakiego typu jest zmienna

Konwersje typów