Contents

[Ohjelmoinnin perusteet PowerShellillä 1](#_Toc73950468)

[Muuttujat 1](#_Toc73950469)

[Muuttujan tyyppi 2](#_Toc73950470)

[Muuttujat Powershell ISE:ssä 3](#_Toc73950471)

[Ehtolauseet 4](#_Toc73950472)

[if else 4](#_Toc73950473)

[Vertailuoperaattorit 6](#_Toc73950474)

[Logiikkaoperaattorit 6](#_Toc73950475)

[Toistolauseet 6](#_Toc73950476)

[foreach 7](#_Toc73950477)

[Syötteen lukeminen 7](#_Toc73950478)

[Dokumentointi 7](#_Toc73950479)

# Ohjelmoinnin perusteet PowerShellillä

Tässä ohjeessa käydään läpi ohjelmoinnin perusteita Windowsin PowerShellillä. Näistä taidoista on hyötyä, kun opettelet ohjelmointia ja lähdet tekemään skriptejä.

# Muuttujat

Muuttujat (variable) ovat ikään kuin säiliöitä, joihin voidaan tallentaa tietoja, kuten numeroita, kirjaimia, sanoja ja komentoja. Muuttujien sisältöä voidaan muokata.

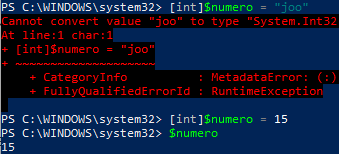
Tässä luodaan muuttuja nimeltään muuttuja, jonka sisältö on ”Tämä on muuttuja”. muuttujan nimen eteen laitetaan aina $-merkki, jolla siitä tehdään muuttuja.

Alla olevalla rivillä tulostetaan $muuttuja sisältö.

Tässä esimerkissä Powershell päätti itse muuttujan sisällön. Eli minkä tyyppinen muuttuja se oli.



Tässä luodaan int eli kokonaisluku tyyppinen muuttuja. Eli muuttujan nimenalkuun laitetaan [] sisälle muuttujan tyyppi. Sille voi siis antaa arvoja, kuten 5, 600 tai 12397.



Arvoksi voi antaa myös desimaaliluvun, mutta todellisuudessa siitäkin tulee kokonaisluku.



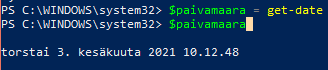
Tässä luodaan double eli liukuluku (desimaaliluku).



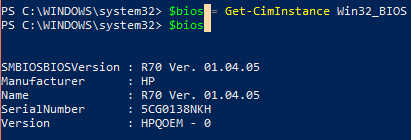
Tässä luodaan string-tyyppinen muuttuja. Stringit voivat sisältää mitä tahansa tekstiä ja numeroita.



Muuttujaan voi myös tallentaa jonkin komennon sisällön. Tässä $paivamaara muuttujana tallennetaan get-date komento. Kun $paivamaara muuttujan sisältöä katsotaan, antaa se päivämäärän.

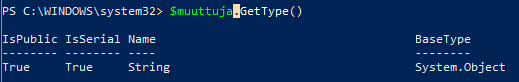


Myös komentojen lisäparametrejä voi talletaa muuttujiin. Tässä $bios muuttujaan tallennetaan tietokoneen BIOS:n tiedot.

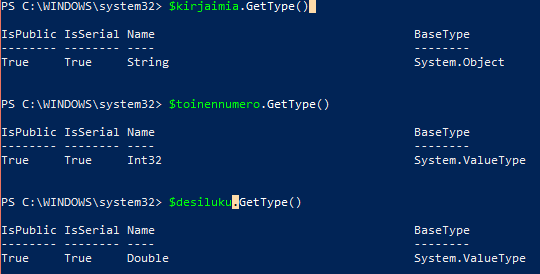


### Muuttujan tyyppi

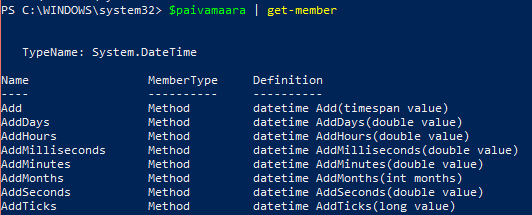
Jos et ole varma minkä tyyppinen muuttuja on, saat sen selville GetType komennolla. Tässä saadaan selville, että muuttuja on tyypiltään String:



String, int ja double.



Jos muuttujan sisältönä on jokin komento, kuten get-date, saadaan sen metodit ja ominaisuudet myös selville get-member komennolla. Get-memberin kautta selville saadut metodit voidaan ottaa käyttöön.



Nyt ei tulosteta koko get-date komennon sisältöä, vaan tulostetaan pelkkä vuosiluku Year-metodilla.



Metodeja voi käyttää Powershell ISE:ssä esimerkiksi näin:

Eli komento sulkuihin ja sen jälkeen . ja metodin nimi, kuten tässä .year.

Jälkimmäisessä kohdassa vuosilukuun ollaan lisätty +3 eli vuosiluku olisikin 2024

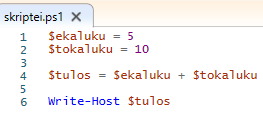




## Muuttujat Powershell ISE:ssä

Powershell ISE on Powershellin ohjelmointiympäristö, jolla tehdään mm. skriptejä. Tässä luodaan kaksi muuttujaa ja annetaan niille arvot.

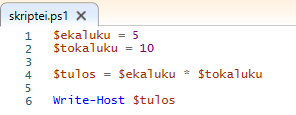
Sitten luodaan uusi muuttuja, joka laskee yhteen kahden aikaisemman muuttujan arvot ja lopuksi tulostaa niiden tuloksen.



Tuloste ISE:ssä.

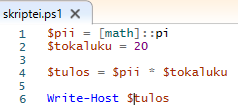


Tässä muutetaan lasku kertolaskuksi.

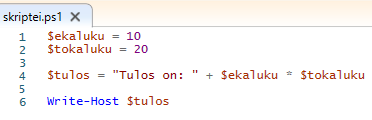




Powershellistä löytyy myös matemaattisia funktiota, esimerkiksi Piin arvo.



Tässä lisätään tulosteeseen vielä tekstiä, joka kertoo, että ”Tulos on: ”.



# Ehtolauseet

Ohjelmoinnissa käytetään paljon ehtolauseita. Näillä saadaan tehtyä valintoja erilaisten asioiden välillä. Esimerkiksi: jos valo on vihreä, voit ajaa, jos valo on punainen, et saa ajaa. Tämä on lyhyt esimerkki ehtolauseesta.

Ehtolauseita voidaan tehdä if ja switch rakenteilla.

## if else

Tässä on esimerkki if-lauseesta ensin vuokaaviona ja sitten koodina. Tässä katsotaan, onko $ekaluku suurempi arvoltaan kuin $tokaluku. Sen mukaan pitääkö tämä paikkaansa, suoritetaan eri koodinpätkä.

JOS (if) luku on suurempi = tee näin.

MUUTEN (else) = tee näin. (eli luku EI olisi suurempi)

Ehto onko ekaluku > tokaluku

if

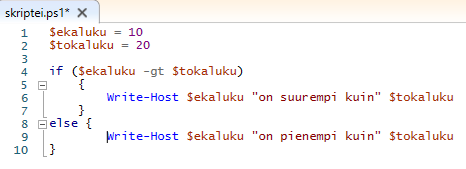
Tulostetaan, että ekaluku on **suurempi**

else

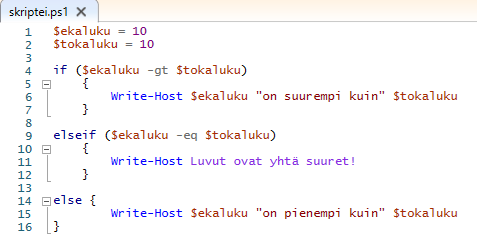
Tulostetaan, että ekaluku on **pienempi**

Ohjelma jatkuu

Tässä määritetään muuttujille ensin arvot. Sitten tehdään if-lauseke, joka katsoo onko ekaluku suurempi kuin tokaluku. Tämä vertailu saadaan aikaan -gt parametrilla eli Greater Than eli Suurempi Kuin vertailuoperaattorilla. Koska ekaluku (arvo 10) on pienempi kuin tokaluku (arvo 20), tulostetaan else-kohdan viesti.



if else lauseeseen voi lisätä myös elseif-kohdan, joka toimii ikään kuin toisena ehtona, jos ensimmäinen if ei pidä paikkaansa. Tässä on lisätty elseif, joka katsoo ovat ekaluku ja tokaluku täysin samat eli Equal eli Yhtä suuret toistensa kanssa. Jos arvot ovat yhtäsuuret, tulostetaan: ”Luvut ovat yhtä suuret!”.



## Vertailuoperaattorit

Ohjelmoinnissa käytetään vertailuoperaattoreita, kun vertaillaan erilaisia arvoja keskenään. Yllä olevissa esimerkeissä käytettiin jo gt ja eq operaattoreita. Esimerkiksi: jotta vesi jäätyy, tule lämpötilan olla Pienempi kuin 0 astetta.

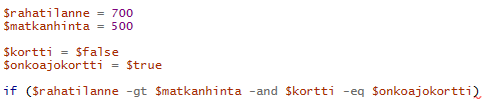
Muita operaattoreita ovat esimerkiksi lt eli Less Than eli pienempi kuin. Sekä ge eli Greater Than or Equal eli Suurempi kuin tai Yhtä suuri. ne eli Not Equal eli Ei Yhtä suuri voi myös olla hyödyllinen.

Täydellinen lista operaattoreista löytyy täältä: <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_comparison_operators?view=powershell-7.1>

## Logiikkaoperaattorit

Logiikkaoperaattoreilla voidaan määritellä tuleeko esimerkiksi kahden eri väittämän pitää paikkaansa tai vain jompikumpi. Esimerkiksi: voit lähteä matkalle jos sinulla on rahaa JA ajokortti.

Tässä ei voida lähteä matkaan. Rahaa olisi koska rahatilanne on suurempi kuin matkanhinta. Mutta ajokorttia ei löydy koska se on false eli epätosi vaikka sen pitäisi olla true eli tosi.



Logiikkaoperaattoreista löytyy lista täältä: <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_logical_operators?view=powershell-7.1>

# Toistolauseet

Toistolausetta toistetaan, kunnes jokin ehto täyttyy. Esimerkiksi: säästä rahaa, kunnes on varaa ostaa 3000€ auto ja sitten lopeta säästäminen.

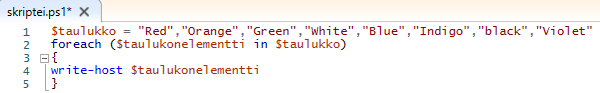
Toistolauseita käytetään paljon erityisesti silloin, kun tietoa on paljon ja sitä halutaan ”loopata” eli toistaen käydä tietoja läpi ja tehdä sille jotakin. Hyvä esimerkki on uusien käyttäjien massaluominen. Silloin jokainen käyttäjä on oma objektinsa ja foreach lauseella käydään läpi jokainen objekti (eli käyttäjä) ja luodaan heille tili. Tulet tekemään tällaisen skriptin seuraavassa PowerShell harjoituksessa.

## foreach

Tässä on luotu taulukko (eli array) nimeltään taulukko. Taulukko on muuttuja, johon voi tallentaa ikään kuin jonoon paljon tietoa. Tässä taulukossa on eri värejä, ja ne on eroteltu toisistaan pilkuilla.

Taulukossa jokainen kohta on oma elementtinsä, jolla on ns. indeksi luku. Indeksi luvut alkavat nollasta. Eli tämän taulukon 0 kohta on Red ja 3 kohta on White.

Foreach looppi käy läpi jokaisen taulukonelementin (alkaen indeksistä 0, loppuen 7), joka taulukossa on. Jokaisella kierroksella komentokehotteeseen kirjoitetaan kyseinen taulukonelementti. Eli ensimmäisellä kierroksella kirjoitetaan indeksi 0 eli Red, seuraavalla kierroksella kirjoitetaan indeksi 1 eli Orange, jne.

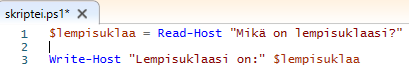


# Syötteen lukeminen

Aikaisemmin tietoja on syötetty muuttujiin valmiiksi heti ohjelman alussa. Tietoja voi kysellä myös käyttäjältä itseltään. Tähän käytetään Read-Host komentoa. Read-Host näyttää ruudulla halutun tekstin, esimerkiksi kysymyksen, ja käyttäjän vastaus tallennetaan muuttujaan.

Tässä lempisuklaa muuttujaan tallennetaan sisältö, joka saadaan, kun käyttäjältä kysytään hänen lempisuklaansa.

Tämän jälkeen hänen lempisuklaansa tulostetaan näytölle.



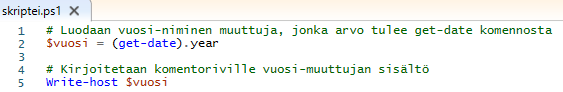
Tältä toiminta näyttää:



# Dokumentointi

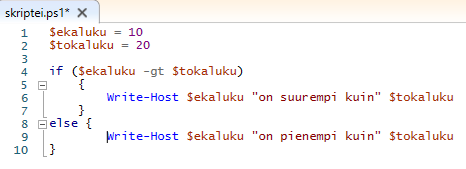
#-merkeillä saat aikaan kommentin.

Skriptien kommentointi on tärkeää, koska niitä käyttävät myös muut kuin sinä. Skriptejä usein myös muokataan, jolloin on tärkeää tietää, miten skripti toimii. Joskus skriptiä voidaan muokata vasta vuosia sen tekemisen jälkeen, jolloin sen toiminta on saattanut unohtua.

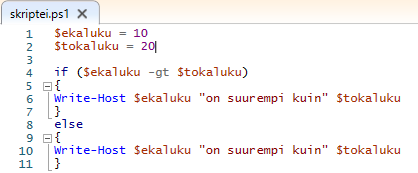


Sisennyksillä koodista saadaan helpommin luettava. Eli kaikki ei ala vasemmalta puolelta ruutua, vaan niitä on sisennetty enemmän oikealla sitä mukaan kun esimerkiksi ehtolauseen komennot etenevät.

Tässä tehty hyvin:



Tässä tehty huonosti, koska sisennystä ei ole.



Tässä myös tehty huonosti, koska kaikki on samalla rivillä. Koodin lukeminen on hankalaa.

