Bash-cygwin

declaring variables

Bash types

In Bash, variables are untyped. If we assign an integer value to a variable, we can perform integer arithmetics on it. We can treat it as an integer variable.

If we assign a string value to a variable, we can perform string manipulating operations on it.

But if we try to perform integer arithmetics on a variable that actually has a string value, the value of the variable is treated as 0 (zero).

```
$ i=abc
$ echo $i
abc
$ echo $((i + 1)) # the string abc is interpreted as value 0
1
```

Using declare

There is, however, a weak form of typing available in Bash. We can use the declare (or typeset) built-in command to declare that a variable has certain properties.

To declare a variable to have some integer properties we can say:

```
$ declare -i my_int
```

Now, Bash can treat my_int as an integer in some cases. But we can still do:

```
$ my_int="abc" # my_int will have value 0
```

Some examples of declare

```
declare -i  var         # var will be treated as integer

declare -r my_const     # my_const will be readonly

declare -a names        # names will be an array

declare -A dictionary   # dictionary will be an associative array
```

Read more:
https://www.gnu.org/software/bash/manual/bashref.html#Bash-Builtins

declare -i

Using declare -i allows for some treatment of a variable as an int:

```
$ declare -i my_int
$ my_int=1+3*4
$ echo $my_int
13
### Compare to:
$ normal_var=1+3*4
$ echo $normal_var
1+3*4
```

declare -r

We may create a readonly variable using declare -r:

```
$ declare -r MAX_SCORE=10
$ MAX_SCORE=11
bash: MAX_SCORE: readonly variable

# MAX_SCORE cannot be re-assigned or even unset
```

declare -a

We can declare that a variable should be considered an array:

```
$ declare -a names
```

Now names will be treated as an array. We may use read -a to interactively set the elements of the array:

```
$ read -a names
Rikard Henrik Jonas                (user enters 3 names and [Enter])
$ declare -p | grep 'names=' # declare -p prints all variables
declare -a names='([0]="Rikard" [1]="Henrik" [2]="Jonas")'
$ echo ${names[1]}
Henrik
```

Read more on arrays: http://www.tldp.org/LDP/abs/html/arrays.html

declare -A

If we need an associative array, we use declare -A

```
$ declare -A name_to_phone_nr
$ name_to_phone_nr[Henrik]=031-12345678
$ name_to_phone_nr[Rikard]=08-666-666
$ echo ${name_to_phone_nr[Rikard]}
08-666-666
```

Associative arrays (like Map or dictionary)

http://www.gnu.org/software/bash/manual/html_node/Arrays.html