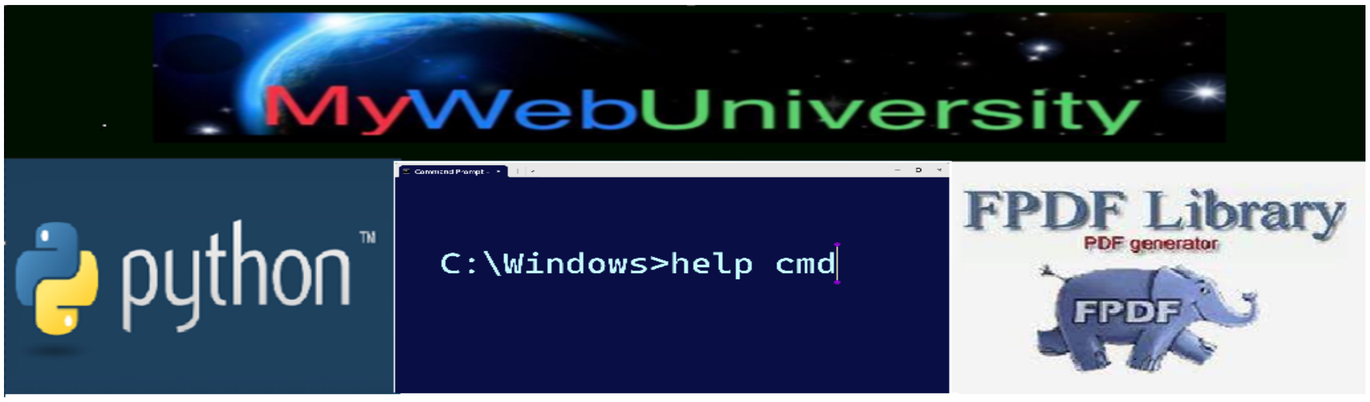

Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Windows 11 Help on 'SET' command

C:\>HELP SET

Displays, sets, or removes cmd.exe environment variables.

SET [variable=[string]]

  variable Specifies the environment-variable name.

  string   Specifies a series of characters to assign to the variable.

Type SET without parameters to display the current environment variables.

If Command Extensions are enabled SET changes as follows:

SET command invoked with just a variable name, no equal sign or value

will display the value of all variables whose prefix matches the name

given to the SET command. For example:

```
    SET P
```

would display all variables that begin with the letter 'P'

SET command will set the ERRORLEVEL to 1 if the variable name is not

found in the current environment.

SET command will not allow an equal sign to be part of the name of

a variable.

Two new switches have been added to the SET command:

```
    SET /A expression
    SET /P variable=[promptString]
```

The /A switch specifies that the string to the right of the equal sign is a numerical expression that is evaluated. The expression evaluator is pretty simple and supports the following operations, in decreasing order of precedence:

```
    ()                  - grouping
    ! ~ -               - unary operators
    * / %               - arithmetic operators
    + -                 - arithmetic operators
    << >>               - logical shift
    &                   - bitwise and
    ^                   - bitwise exclusive or
    |                   - bitwise or
    = *= /= %= += -=    - assignment
      &= ^= |= <<= >>=
    ,                   - expression separator
```

If you use any of the logical or modulus operators, you will need to enclose the expression string in quotes. Any non-numeric strings in the expression are treated as environment variable names whose values are converted to numbers before using them. If an environment variable name is specified but is not defined in the current environment, then a value of zero is used. This allows you to do arithmetic with environment variable values without having to type all those % signs to get their values. If SET /A is executed from the command line outside of a command script, then it displays the final value of the expression. The assignment operator requires an environment variable name to the left of

the assignment operator. Numeric values are decimal numbers, unless
prefixed by 0x for hexadecimal numbers, and 0 for octal numbers.
So 0x12 is the same as 18 is the same as 022. Please note that the octal
notation can be confusing: 08 and 09 are not valid numbers because 8 and
9 are not valid octal digits.

The /P switch allows you to set the value of a variable to a line of input
entered by the user. Displays the specified promptString before reading
the line of input. The promptString can be empty.

Environment variable substitution has been enhanced as follows:

```
    %PATH:str1=str2%
```

would expand the PATH environment variable, substituting each occurrence
of "str1" in the expanded result with "str2". "str2" can be the empty
string to effectively delete all occurrences of "str1" from the expanded
output. "str1" can begin with an asterisk, in which case it will match
everything from the beginning of the expanded output to the first
occurrence of the remaining portion of str1.

May also specify substrings for an expansion.

```
    %PATH:~10,5%
```

would expand the PATH environment variable, and then use only the 5
characters that begin at the 11th (offset 10) character of the expanded
result. If the length is not specified, then it defaults to the
remainder of the variable value. If either number (offset or length) is
negative, then the number used is the length of the environment variable
value added to the offset or length specified.

```
    %PATH:~-10%
```

would extract the last 10 characters of the PATH variable.

```
    %PATH:~0,-2%
```

would extract all but the last 2 characters of the PATH variable.

Finally, support for delayed environment variable expansion has been added. This support is always disabled by default, but may be enabled/disabled via the /V command line switch to CMD.EXE. See CMD /?

Delayed environment variable expansion is useful for getting around the limitations of the current expansion which happens when a line of text is read, not when it is executed. The following example demonstrates the problem with immediate variable expansion:

```
    set VAR=before
    if "%VAR%" == "before" (
        set VAR=after
        if "%VAR%" == "after" @echo If you see this, it worked
    )
```

would never display the message, since the %VAR% in BOTH IF statements is substituted when the first IF statement is read, since it logically includes the body of the IF, which is a compound statement. So the IF inside the compound statement is really comparing "before" with "after" which will never be equal. Similarly, the following example will not work as expected:

```
    set LIST=
    for %i in (*) do set LIST=%LIST% %i
    echo %LIST%
```

in that it will NOT build up a list of files in the current directory,
but instead will just set the LIST variable to the last file found.
Again, this is because the %LIST% is expanded just once when the
FOR statement is read, and at that time the LIST variable is empty.
So the actual FOR loop we are executing is:

```
    for %i in (*) do set LIST= %i
```

which just keeps setting LIST to the last file found.

Delayed environment variable expansion allows you to use a different
character (the exclamation mark) to expand environment variables at
execution time. If delayed variable expansion is enabled, the above
examples could be written as follows to work as intended:

```
    set VAR=before
    if "%VAR%" == "before" (
        set VAR=after
        if "!VAR!" == "after" @echo If you see this, it worked
    )

    set LIST=
    for %i in (*) do set LIST=!LIST! %i
    echo %LIST%
```

If Command Extensions are enabled, then there are several dynamic
environment variables that can be expanded but which don't show up in
the list of variables displayed by SET. These variable values are
computed dynamically each time the value of the variable is expanded.
If the user explicitly defines a variable with one of these names, then
that definition will override the dynamic one described below:

%CD% - expands to the current directory string.

%DATE% - expands to current date using same format as DATE command.

%TIME% - expands to current time using same format as TIME command.

%RANDOM% - expands to a random decimal number between 0 and 32767.

%ERRORLEVEL% - expands to the current ERRORLEVEL value

%CMDEXTVERSION% - expands to the current Command Processor Extensions
    version number.

%CMDCMDLINE% - expands to the original command line that invoked the
    Command Processor.

%HIGHESTNUMANODENUMBER% - expands to the highest NUMA node number
    on this machine.