You probably heard something about the environment relating to the shell. Well, that's some specific bunch of data that helps the shell to work with different programs. Let's explore today such parts of the environment as variables and aliases, parts which are common to many UNIX shells and can be customized for your convenience.

Shell variables

Like a variable in any programming language, a shell variable can also be expressed as a container storing some information/value. Those values are used by programs in the shell to modify their actions. Bash uses many shell variables with a default value assigned to them. For example, the variable

`HOME` is filled with the current user's home directory. It is used as the default for the `cd` command and also when the `~` sign is used in the path to the file. Let's see what is inside this variable using `echo` and the name of the variable with the `$` sign in front of it:

```
$ echo $HOME
```

```
/home/<user name>
```

A subset of shell variables exported to any new shell from the current one is called environment variables. You can convert any variable to an environment variable by using the `export` command:

```
export VARIABLE
```

You can see the list of shell variables by typing the following command:

```
set | less
```

Also, you can view the list of environment variables by using the command below:

```
printenv | less
```

Now, let's take a look at how to create variables in the shell.

Creating variables

You can create your own variable in the shell and use it in scripts like this:

```
$ MY_VARIABLE="some value"
```

```
$ echo $MY_VARIABLE
```

```
some value
```

Just remember when you set the value, you don't need to use the `$` sign, and no spaces around the `=` sign. Also, the name can consist of letters, numbers, and underscore characters, but the first symbol cannot be a number.

To unset the value use `unset` without the `$` sign. Otherwise, the command will refer to the value of the variable.

```
$ unset MY_VARIABLE
```

Variables created this way in the shell will be available only in the current shell. You can save them to one of the start-up files like */etc/profile* or *~/.bashrc* or *~/.bash_profile,* so you can use them every time you start the shell.

PATH of programs

One of the must-know variables is

`PATH`. Actually, it's a list of directories in which a program will be searched. It is usually stored in one of the start-up files.

If you look at what this variable contains in your system, you'll find something like this:

```
$ echo $PATH
```

```
PATH=/home/hagrid/.local/bin: /bin: /usr/bin: /usr/local/bin:
/sbin: /usr/sbin: /usr/local/sbin
```

Likely your result will be a little bit different, as it depends on distribution.

You can add a directory with your programs to the `PATH` and then export variable to the environment like this:

```
PATH=$PATH:/path-to-your-directory
```

```
export PATH
```

If you want to keep it, you need to put those lines in one of your start-up files.

Aliases

Aliases are synonyms or shortcuts for commands that you can define for your convenience. For example, you can rename commands or type something shorter instead of the command with many options.

The `alias` command is used like this:

```
alias name-of-alias='command'
```

Be aware, there shouldn't be any white spaces around the `=` sign.

Let's set an alias for the `rm` command, so it will always be in an interactive mode:

```
alias rm="rm -i"
```

Remember, this will not warn you from `rm -rf /` though (Don't try it!), because option `-f` overrides `-i`.

To watch all predefined and set by you aliases on your system type

`alias` without arguments:

```
$ alias

alias cp='cp -i'
alias df='df -h'
alias egrep='egrep --colour=auto'
alias fgrep='fgrep --colour=auto'
alias free='free -m'
alias grep='grep --colour=auto'
alias ls='ls --color=auto'
alias more='less'
alias np='nano -w PKGBUILD'

alias rm="rm -i"
```

To remove an alias from the environment, use the `unalias` command:

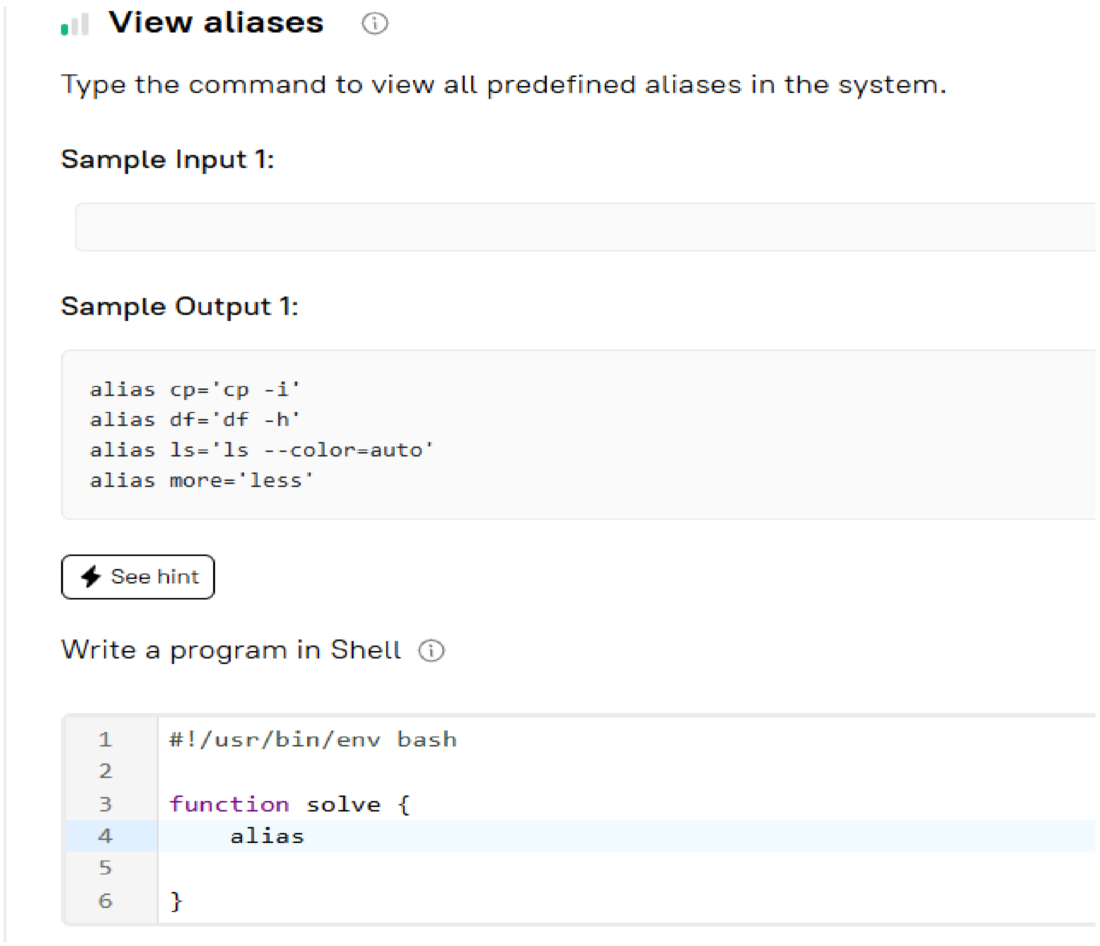
```
unalias rm
```

Again, as with variables, if you want to save an alias to use in another session, you should place it in one of your start-up files.

Conclusion

Let's sum up what we have learned so far:

- shell and environment variables are an important part of the environment, along with aliases; shell variables can become environment ones with the `export` Command; to create variables in the shell follow the `VARIABLE="value"`template, and to use them type the `$` sign in front of their name – `$VARIABLE`; variable `$PATH` stores directories in which your programs will be searched, it can be updated by adding new ones; aliases are simply synonyms of commands for convenience, they are set with the `alias` command.



Type the command to view all predefined aliases in the system.

Sample Input 1:

Sample Output 1:

```
alias cp='cp -i'
alias df='df -h'
alias ls='ls --color=auto'
alias more='less'
```

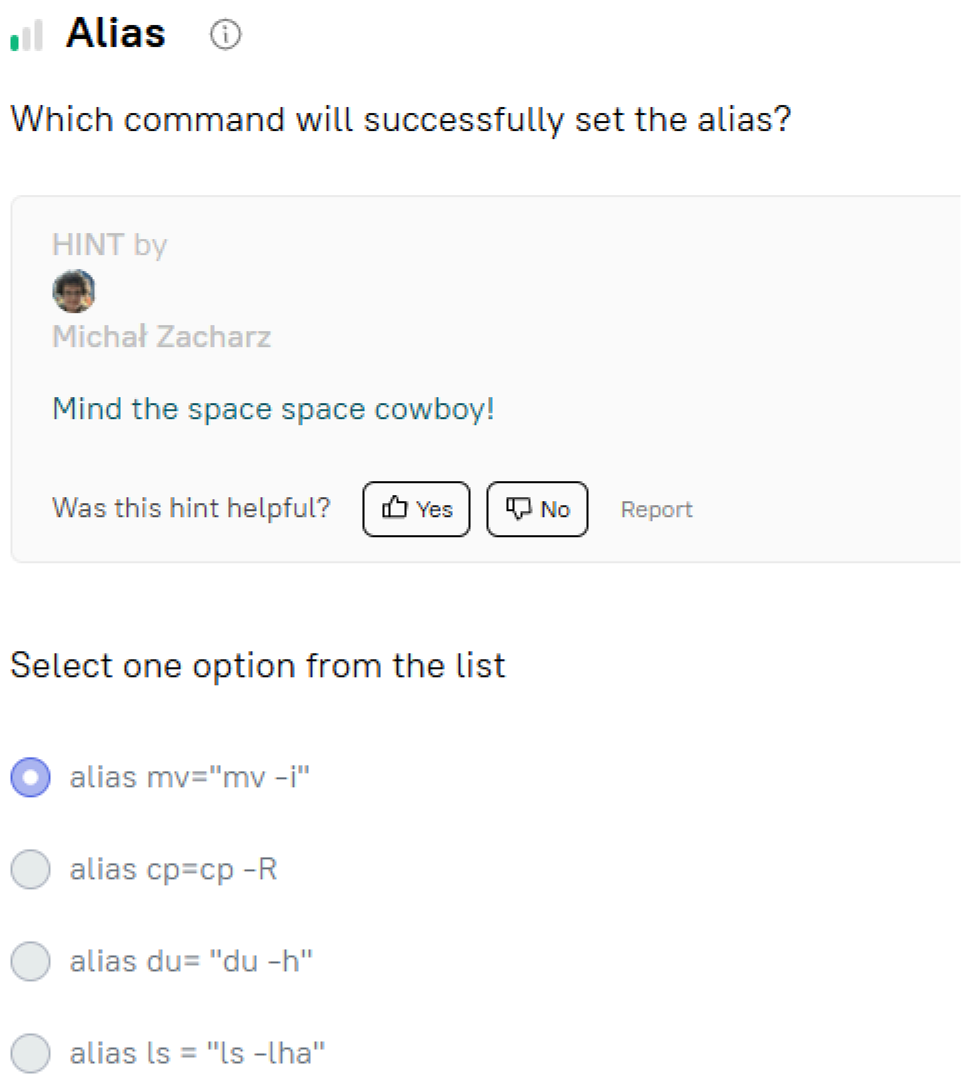
Write a program in Shell

```
#!/usr/bin/env bash

function solve {
    alias

}
```

Which command will successfully set the alias?



Alias

Which command will successfully set the alias?

HINT by

Michał Zacharz

Mind the space space cowboy!

Select one option from the list

- [✓ alias mv="mv -i"]
- alias cp=cp -R
- alias du= "du -h"
- alias ls = "ls -lha"

Manipulation with PATH

You need to append (add, not replace) directory */home/dumbledore/local* to the `PATH` variable.

Note: don't use spaces around the directory path.

```
#!/usr/bin/env bash
function solve {
  PATH=$PATH:/home/dumbledore/local
}
```

Create an alias

Report a typo

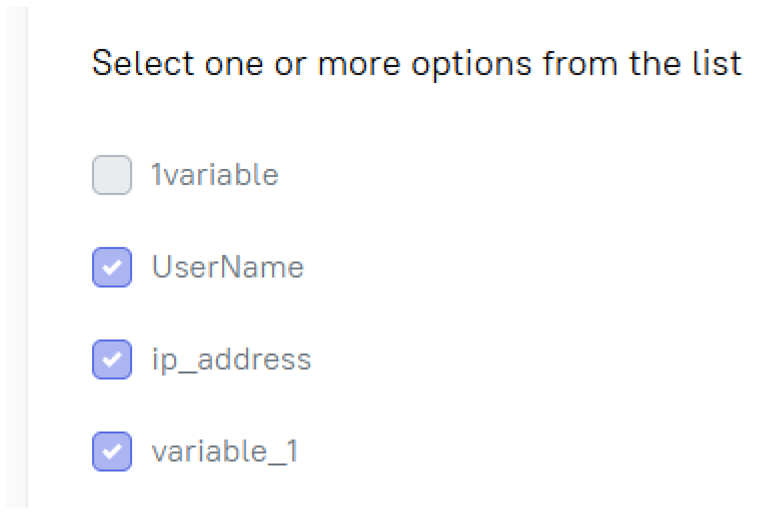
Create an alias under the name `lsl` of `ls` command with the `-l` option ("long listing format").

```
#!/usr/bin/env bash

function solve {
   alias lsl="ls -l"

}
```

Select the correct variable names.



Select one or more options from the list

- 1variable
- [✓ UserName]
- [✓ ip_address]
- [✓ variable_1]