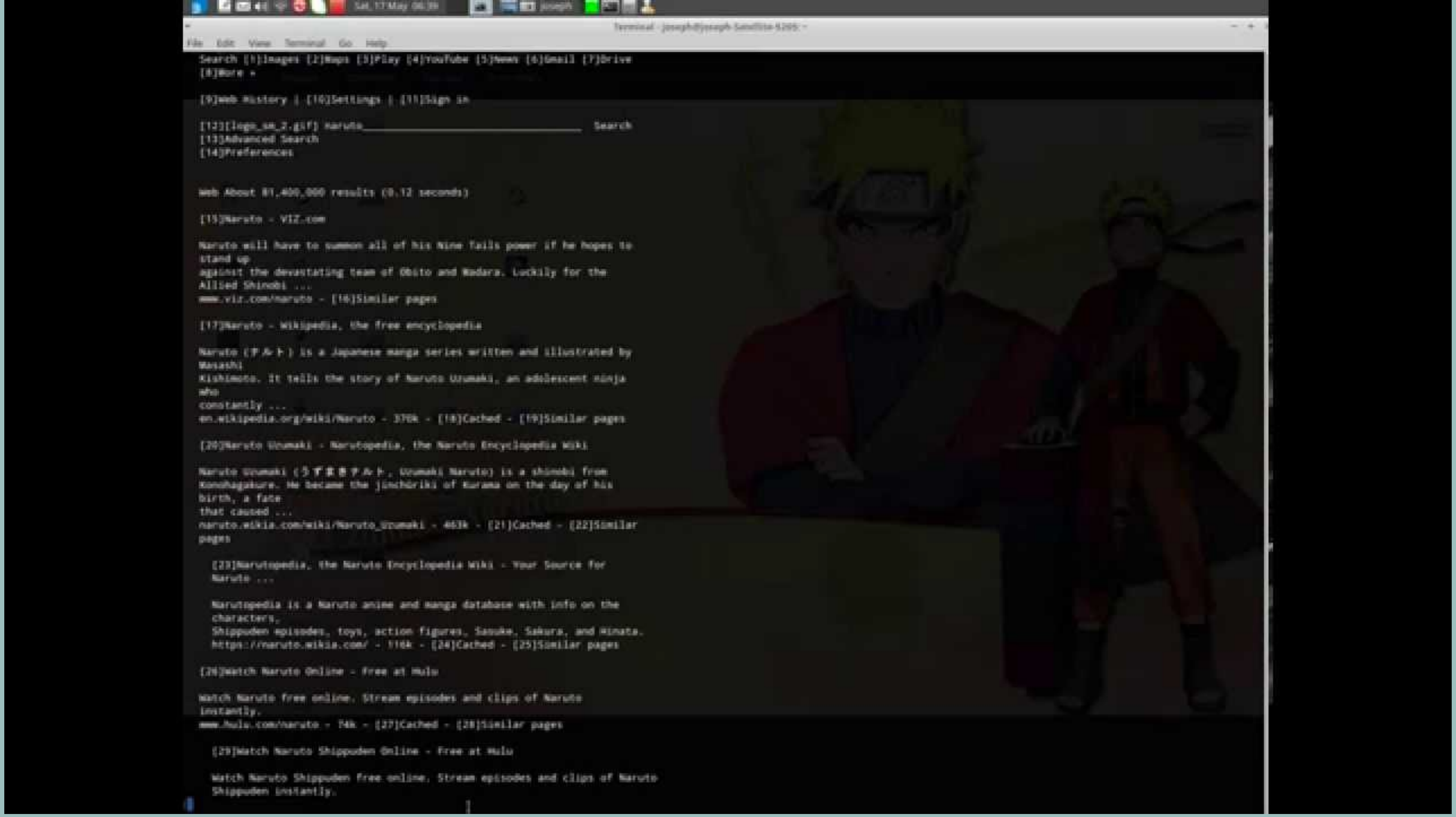


```
Search [1]Images [2]Maps [3]Play [4]YouTube [5]News [6]Gmail [7]Drive
[8]More »

[9]Web History | [10]Settings | [11]Sign in

[12][logo_sm_2.gif] naruto________________________________  Search
[13]Advanced Search
[14]Preferences


Web About 81,400,000 results (0.12 seconds)

[15]Naruto - VIZ.com

Naruto will have to summon all of his Nine Tails power if he hopes to
stand up
against the devastating team of Obito and Madara. Luckily for the
Allied Shinobi ...
www.viz.com/naruto - [16]Similar pages

[17]Naruto - Wikipedia, the free encyclopedia

Naruto (ナルト) is a Japanese manga series written and illustrated by
Masashi
Kishimoto. It tells the story of Naruto Uzumaki, an adolescent ninja
who
constantly ...
en.wikipedia.org/wiki/Naruto - 370k - [18]Cached - [19]Similar pages

[20]Naruto Uzumaki - Narutopedia, the Naruto Encyclopedia Wiki

Naruto Uzumaki (うずまきナルト, Uzumaki Naruto) is a shinobi from
Konohagakure. He became the jinchūriki of Kurama on the day of his
birth, a fate
that caused ...
naruto.wikia.com/wiki/Naruto_Uzumaki - 463k - [21]Cached - [22]Similar
pages

  [23]Narutopedia, the Naruto Encyclopedia Wiki - Your Source for
  Naruto ...

  Narutopedia is a Naruto anime and manga database with info on the
  characters,
  Shippuden episodes, toys, action figures, Sasuke, Sakura, and Hinata.
  https://naruto.wikia.com/ - 116k - [24]Cached - [25]Similar pages

[26]Watch Naruto Online - Free at Hulu

Watch Naruto free online. Stream episodes and clips of Naruto
instantly.
www.hulu.com/naruto - 74k - [27]Cached - [28]Similar pages

  [29]Watch Naruto Shippuden Online - Free at Hulu

  Watch Naruto Shippuden free online. Stream episodes and clips of Naruto
  Shippuden instantly.
```

ATMS 305 WEEK 3: INTRODUCTION TO UNIX/LINUX

Lecture 2: more bash scripting (yay!)

Adapted from: http://ryanstutorials.net/bash-scripting-tutorial/bash-script.php

MORE BASH SCRIPTING TOOLS AND TIPS

Today, we will introduce some more advanced bash scripting techniques that you can use to make scripts more powerful.

INTERACTIVITY

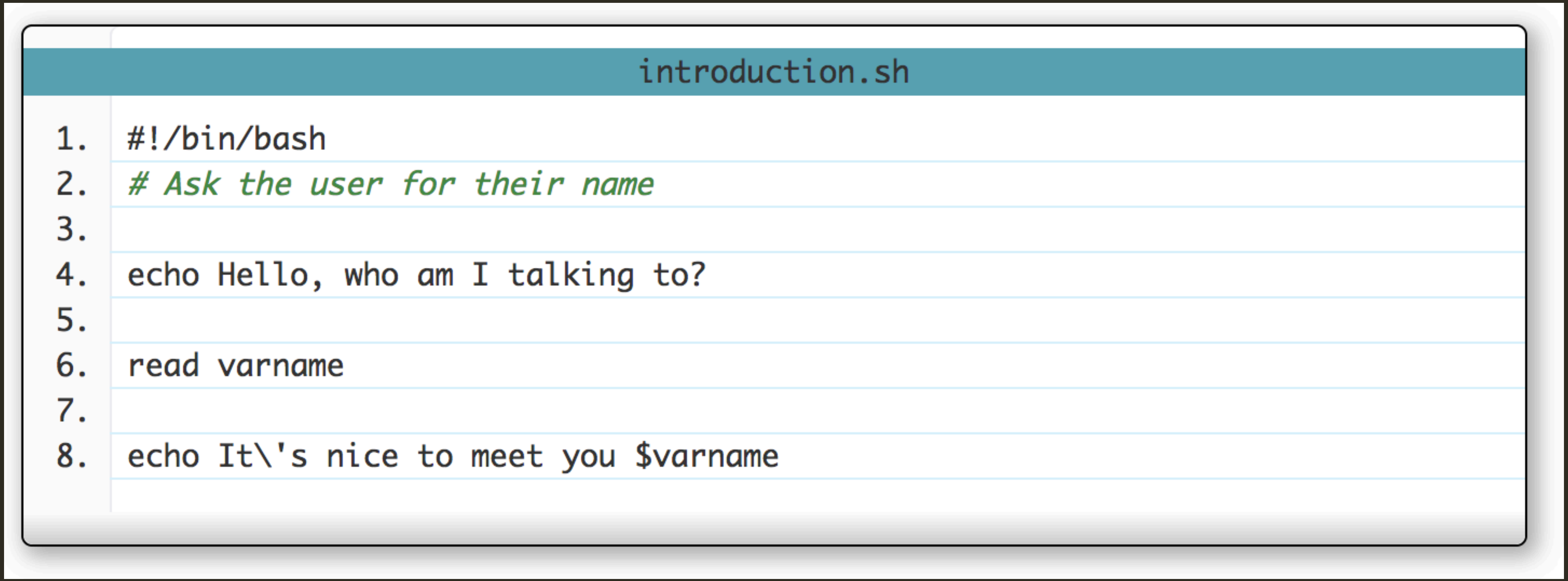
Interactive scripts ask the user for input, and then use that input in the script for something.

In bash, it is easy to generate a prompt in a script.

The read command does the job.

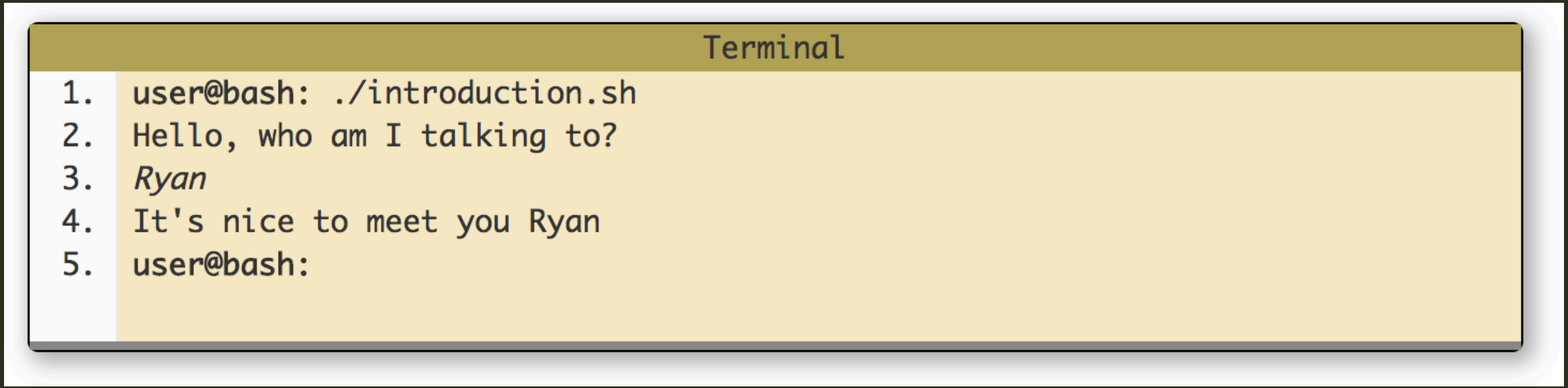
```
read varname
```

will insert the user input into the variable name given



introduction.sh

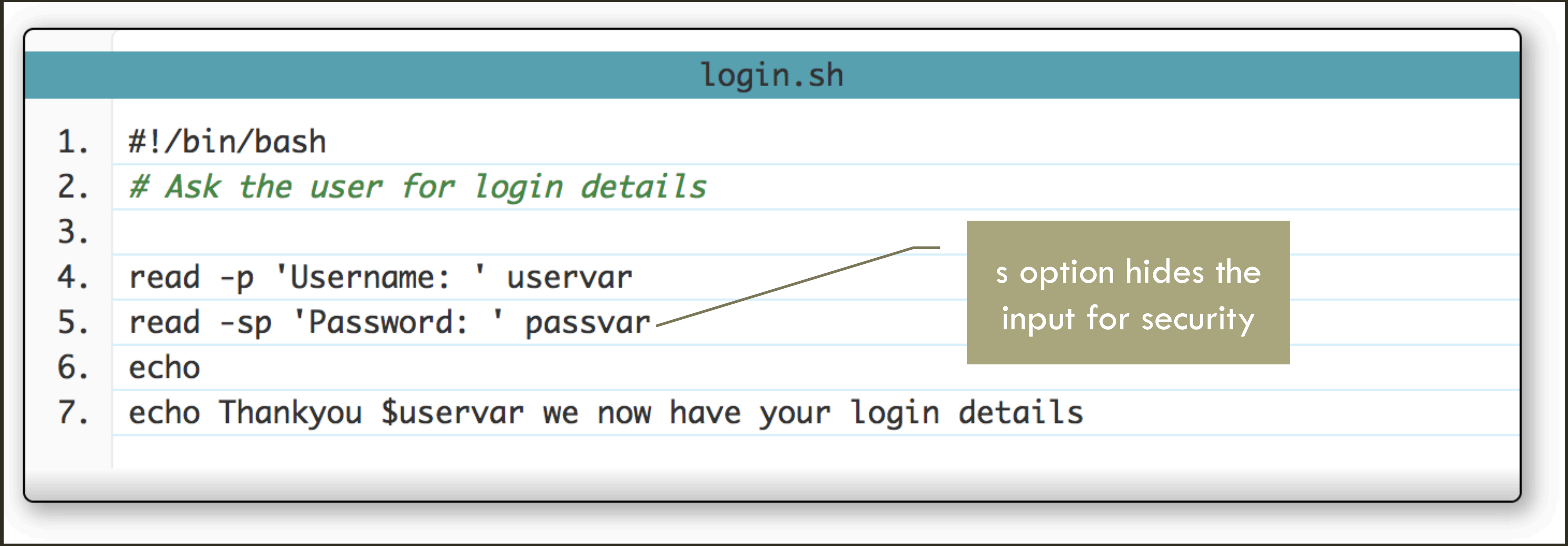
```bash
#!/bin/bash
# Ask the user for their name

echo Hello, who am I talking to?

read varname

echo It\'s nice to meet you $varname
```



Terminal

```
user@bash: ./introduction.sh
Hello, who am I talking to?
Ryan
It's nice to meet you Ryan
user@bash:
```

FANCIER READ COMMANDS

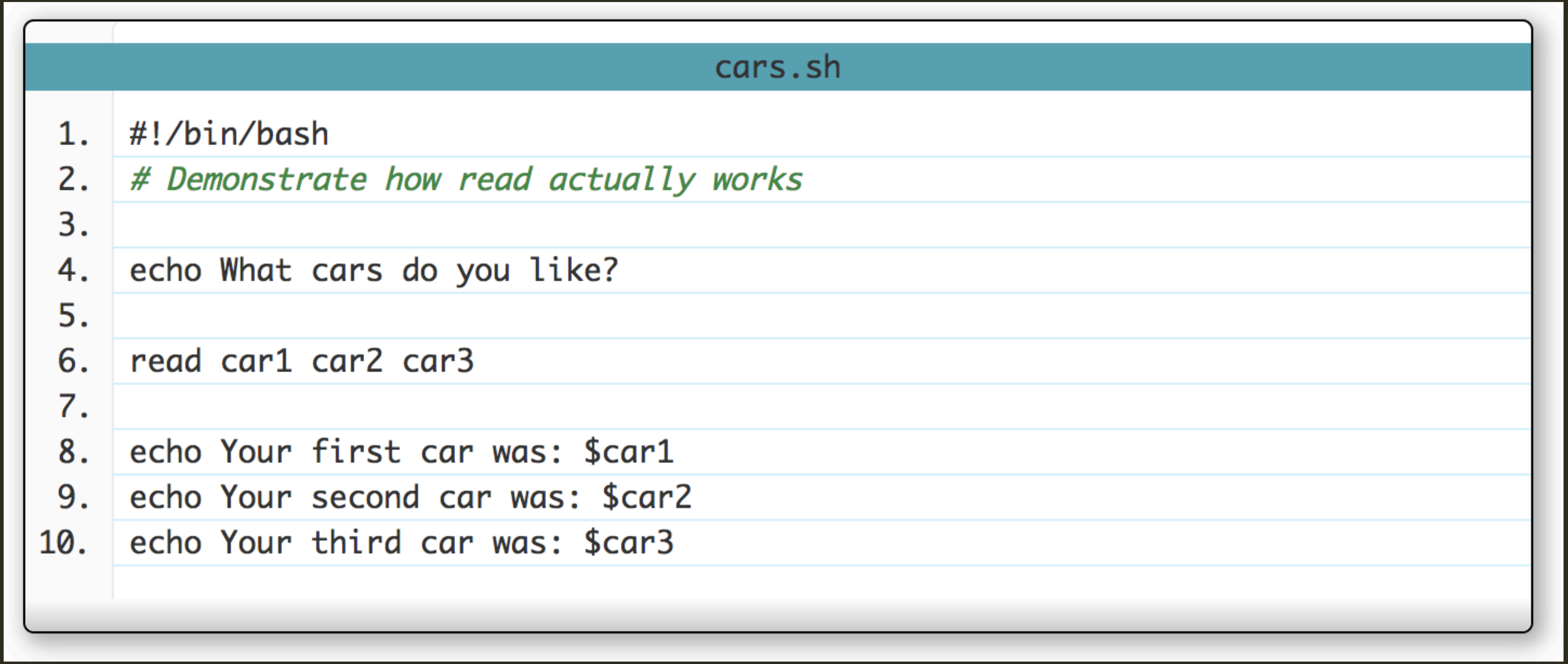


login.sh

```
#!/bin/bash
# Ask the user for login details

read -p 'Username: ' uservar
read -sp 'Password: ' passvar
echo
echo Thankyou $uservar we now have your login details
```

s option hides the input for security

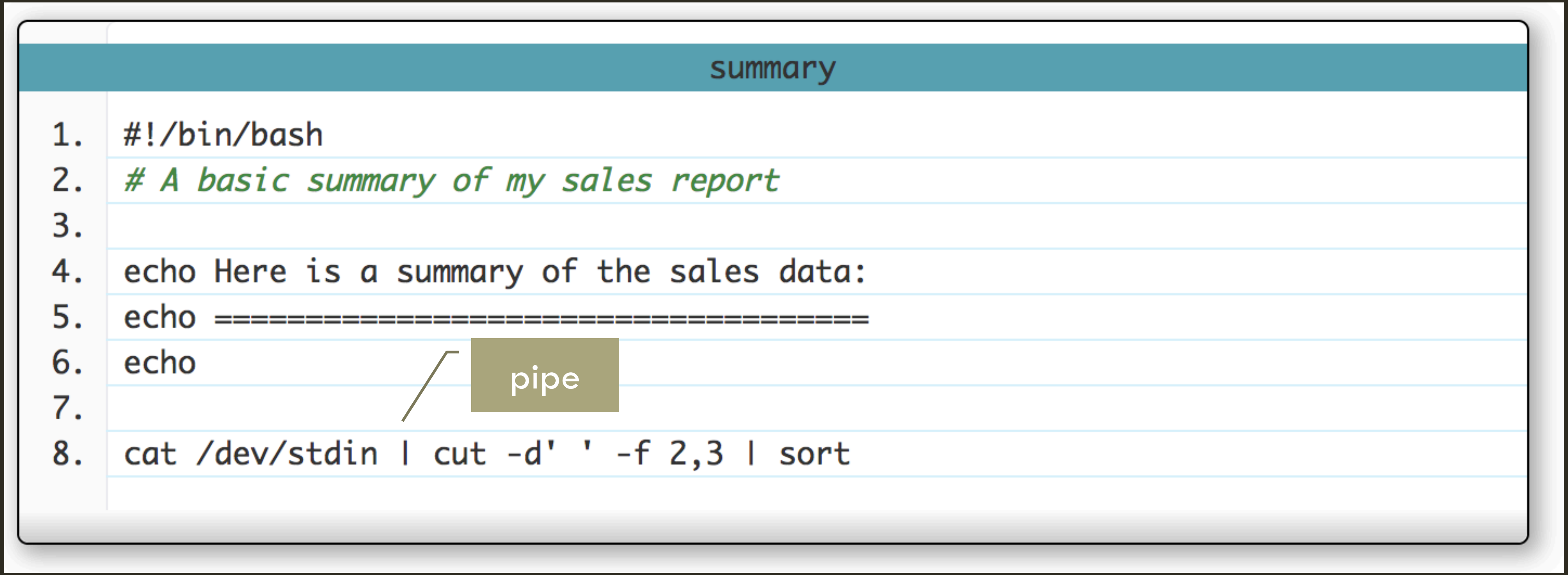
READING MORE THAN ONE VARIABLE AT A TIME


cars.sh

```bash
#!/bin/bash
# Demonstrate how read actually works

echo What cars do you like?

read car1 car2 car3

echo Your first car was: $car1
echo Your second car was: $car2
echo Your third car was: $car3
```

STANDARD INPUT

Standard input is another way to pass information into a script. It uses part of the operating system, and can be used send information in and out of commands and scripts. It is accessible in the filesystem at the location `/dev/stdin`.



summary

```
#!/bin/bash
# A basic summary of my sales report

echo Here is a summary of the sales data:
echo ====================================
echo

cat /dev/stdin | cut -d' ' -f 2,3 | sort
```

summary

```bash
#!/bin/bash
# A basic summary of my sales report

echo Here is a summary of the sales data:
echo ====================================
echo

cat /dev/stdin | cut -d' ' -f 2,3 | sort
```

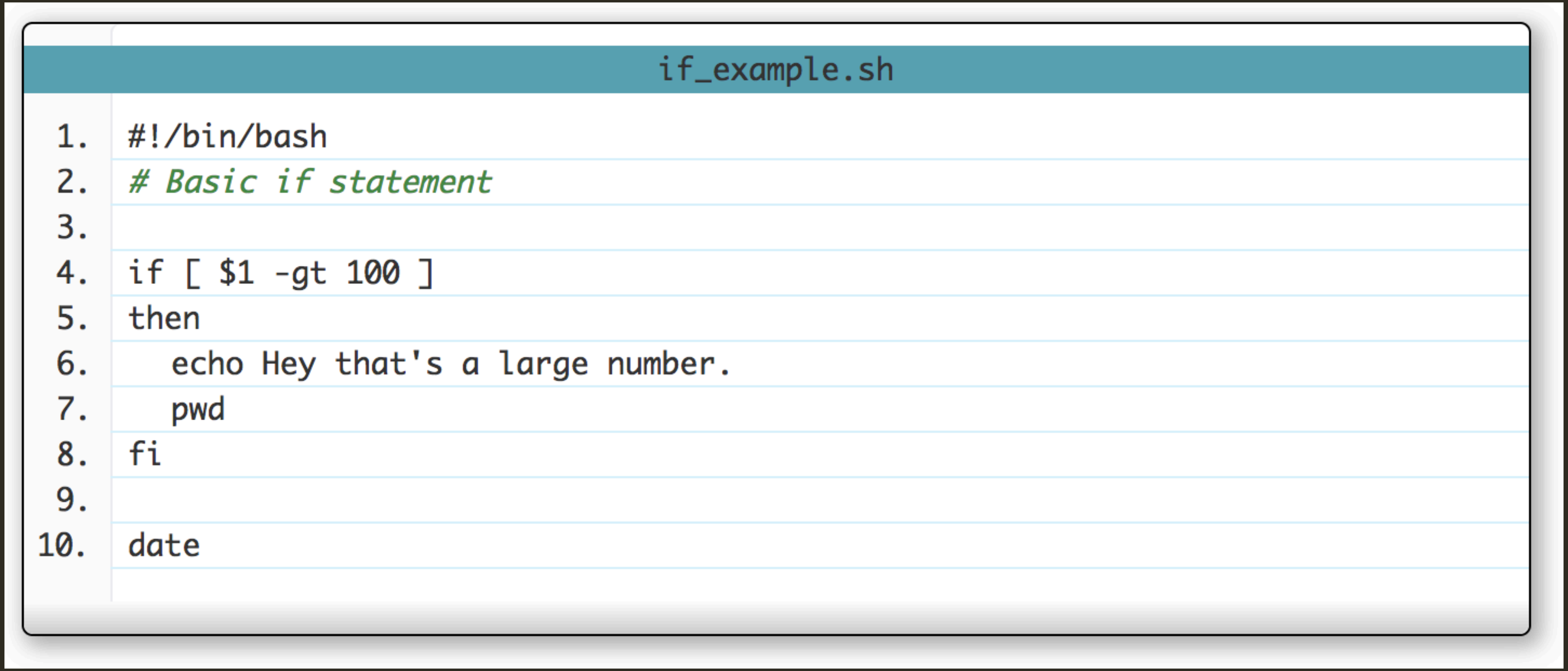
Terminal

```
user@bash: cat salesdata.txt
Fred apples 20 February 4
Susy oranges 5 February 7
Mark watermelons 12 February 10
Terry peaches 7 February 15
user@bash:
user@bash: cat salesdata.txt | ./summary
Here is a summary of the sales data:
====================================

apples 20
oranges 5
peaches 7
watermelons 12
```

INPUT INTO SCRIPTS

So we now have 3 methods for getting input from the user:

- Command line arguments ($1, $2, $3, etc.)
- Read input during script execution (read varname1)
- Accept data that has been redirected into the Bash script via STDIN



if_example.sh

```bash
#!/bin/bash
# Basic if statement

if [ $1 -gt 100 ]
then
  echo Hey that's a large number.
  pwd
fi

date
```

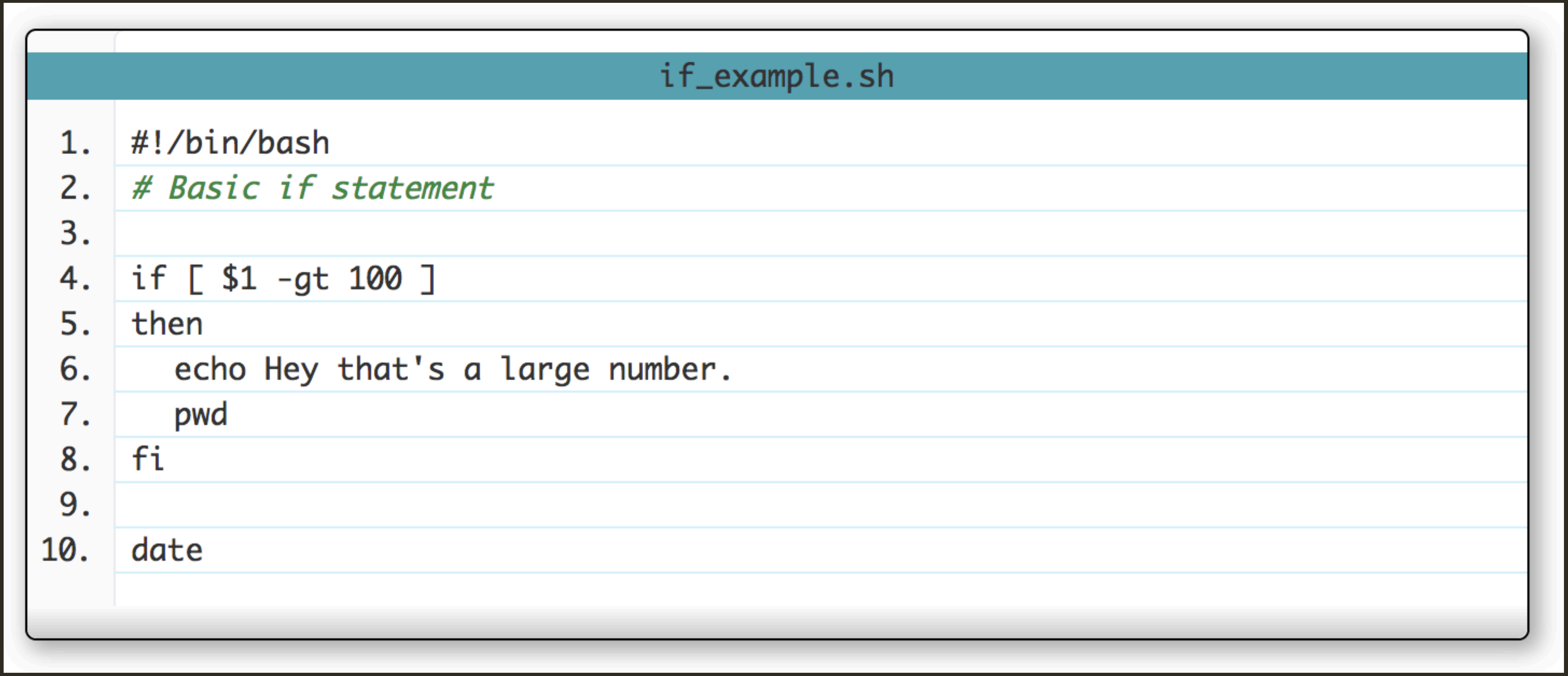
CONTROL STATEMENTS

Scripts can be set up to perform larger tasks using control statements that allow for decisions to be made based on values of variables or specified numbers of repetitions.

This is done with statements such as

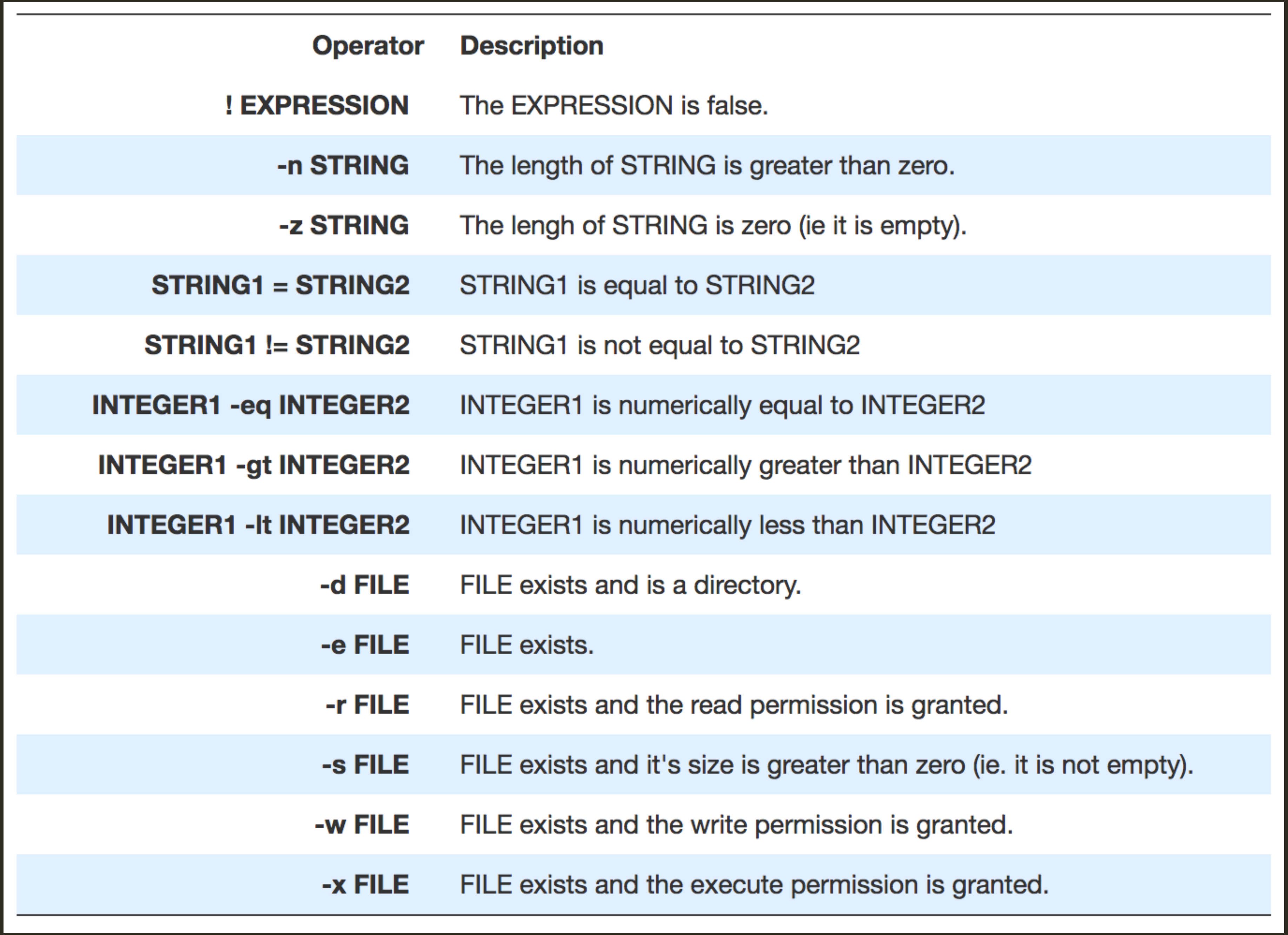
```
if/then/else
while
for
```

IF/THEN SYNTAX

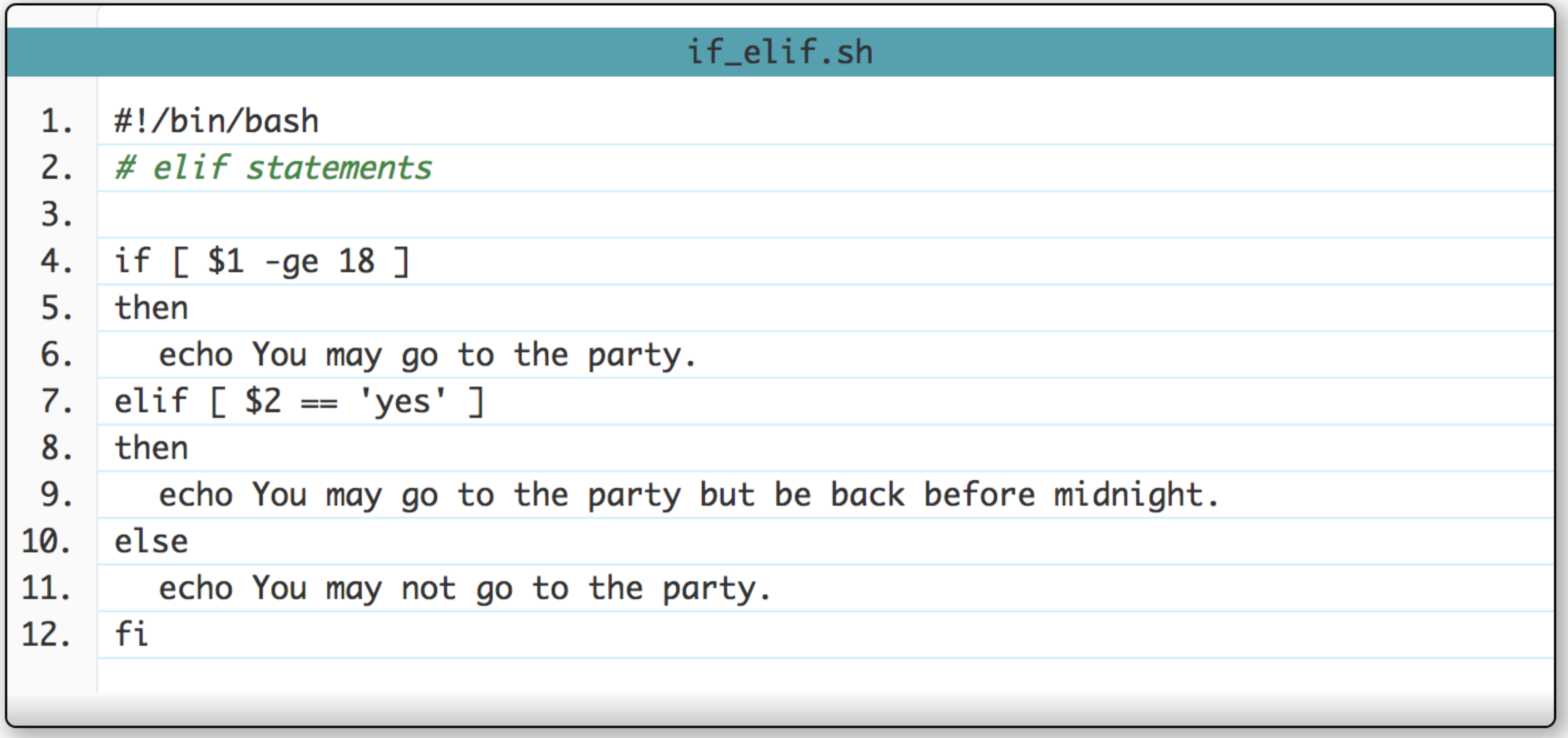


```bash
#!/bin/bash
# Basic if statement

if [ $1 -gt 100 ]
then
   echo Hey that's a large number.
   pwd
fi

date
```

OPERATORS USEFUL IN BASH IF STATEMENTS



Operator	Description
! EXPRESSION	The EXPRESSION is false.
-n STRING	The length of STRING is greater than zero.
-z STRING	The lengh of STRING is zero (ie it is empty).
STRING1 = STRING2	STRING1 is equal to STRING2
STRING1 != STRING2	STRING1 is not equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is numerically equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is numerically greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is numerically less than INTEGER2
-d FILE	FILE exists and is a directory.
-e FILE	FILE exists.
-r FILE	FILE exists and the read permission is granted.
-s FILE	FILE exists and it's size is greater than zero (ie. it is not empty).
-w FILE	FILE exists and the write permission is granted.
-x FILE	FILE exists and the execute permission is granted.

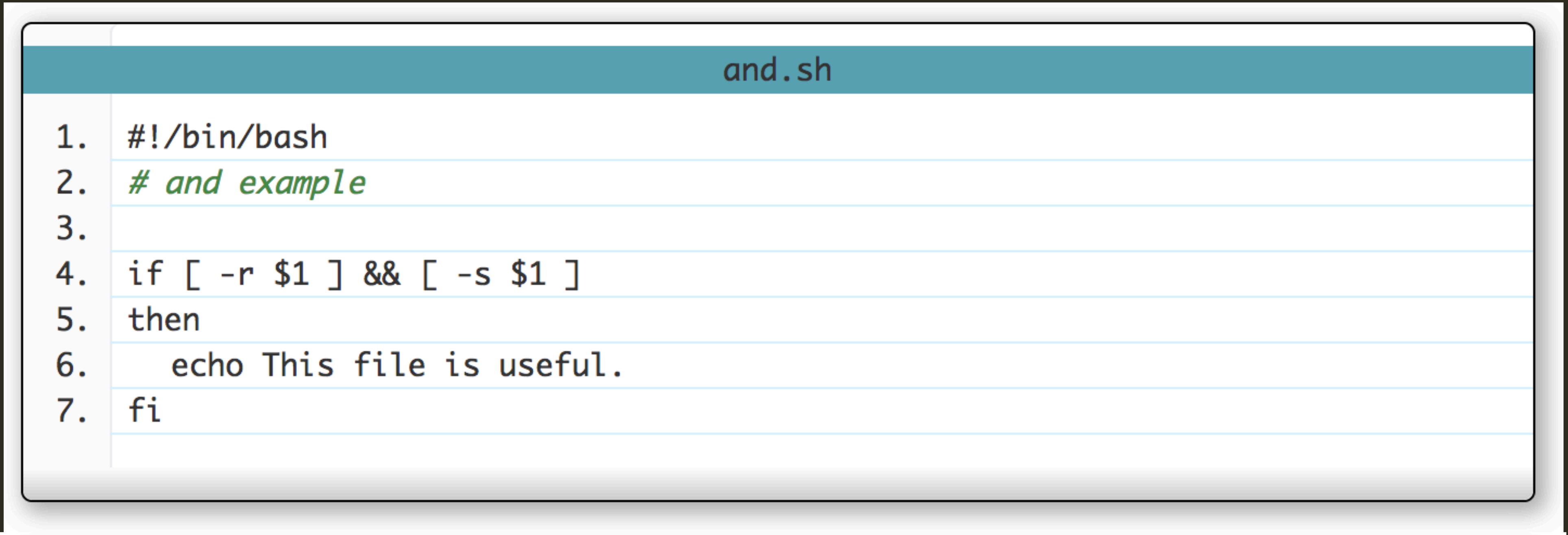


if_elif.sh

```bash
#!/bin/bash
# elif statements

if [ $1 -ge 18 ]
then
  echo You may go to the party.
elif [ $2 == 'yes' ]
then
  echo You may go to the party but be back before midnight.
else
  echo You may not go to the party.
fi
```

MULTIPLE CONDITIONS

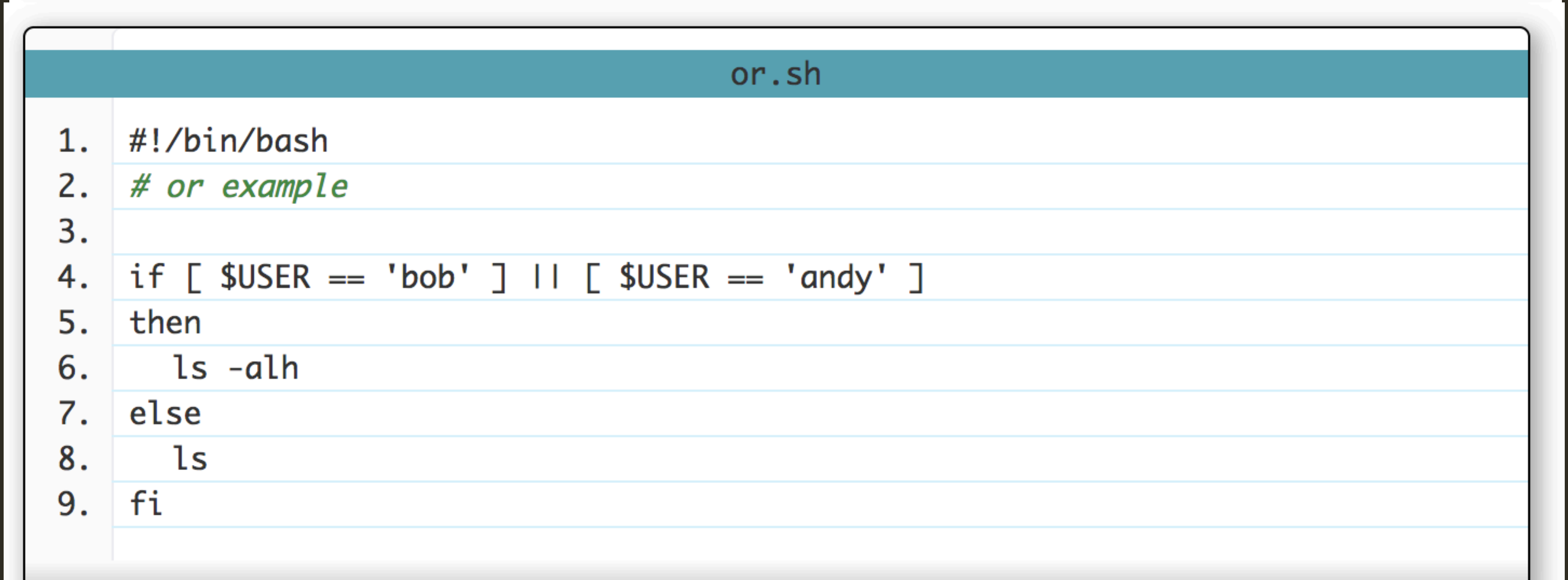
AND &&



and.sh

```bash
#!/bin/bash
# and example

if [ -r $1 ] && [ -s $1 ]
then
   echo This file is useful.
fi
```

OR | |

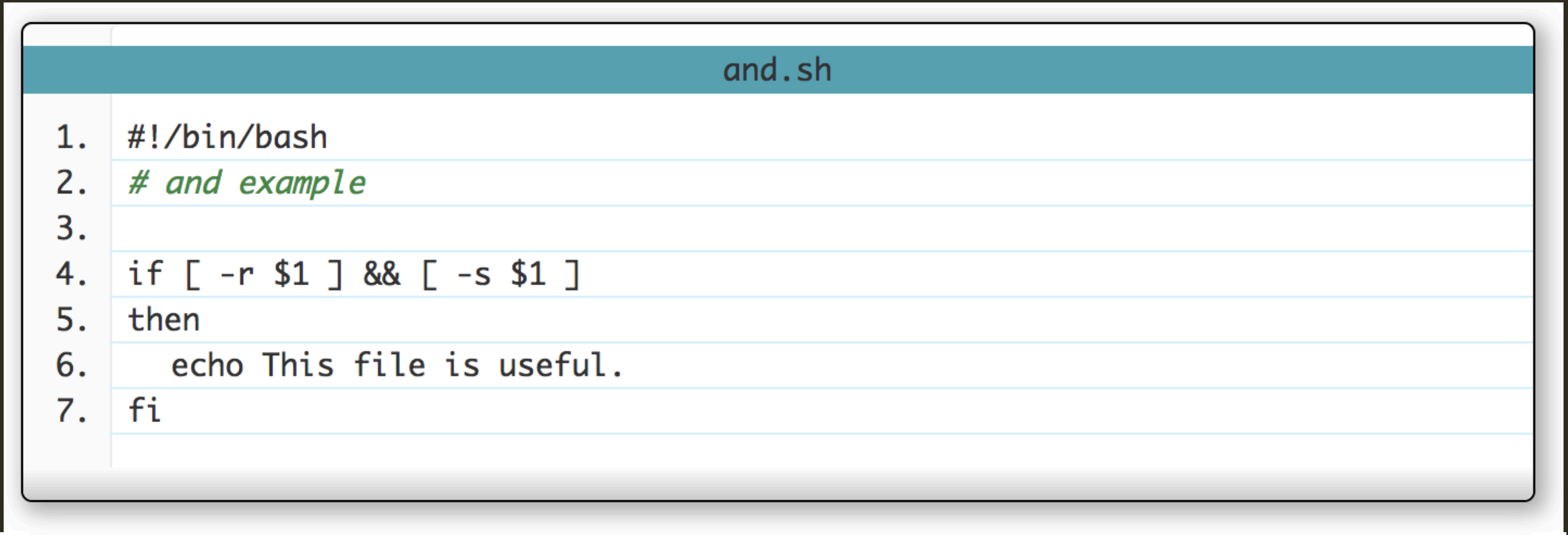


or.sh

```bash
#!/bin/bash
# or example

if [ $USER == 'bob' ] || [ $USER == 'andy' ]
then
   ls -alh
else
   ls
fi
```

MULTIPLE CONDITIONS

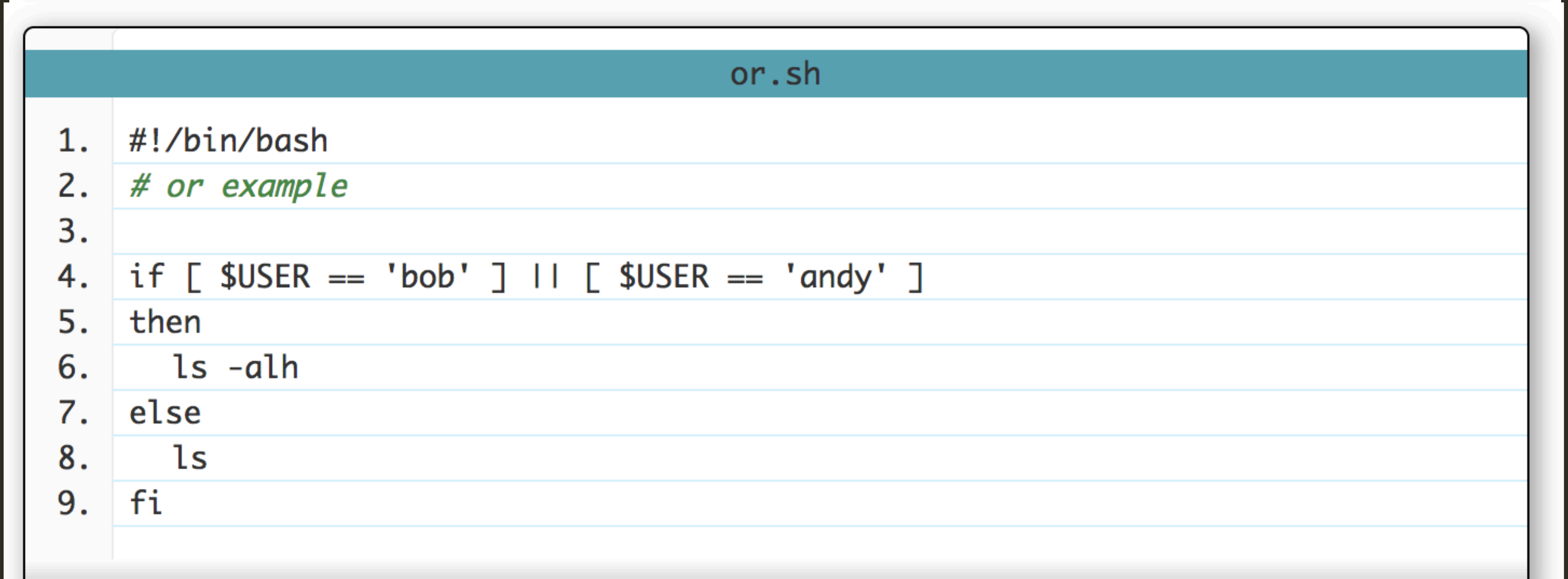
AND &&



and.sh

```
#!/bin/bash
# and example

if [ -r $1 ] && [ -s $1 ]
then
  echo This file is useful.
fi
```

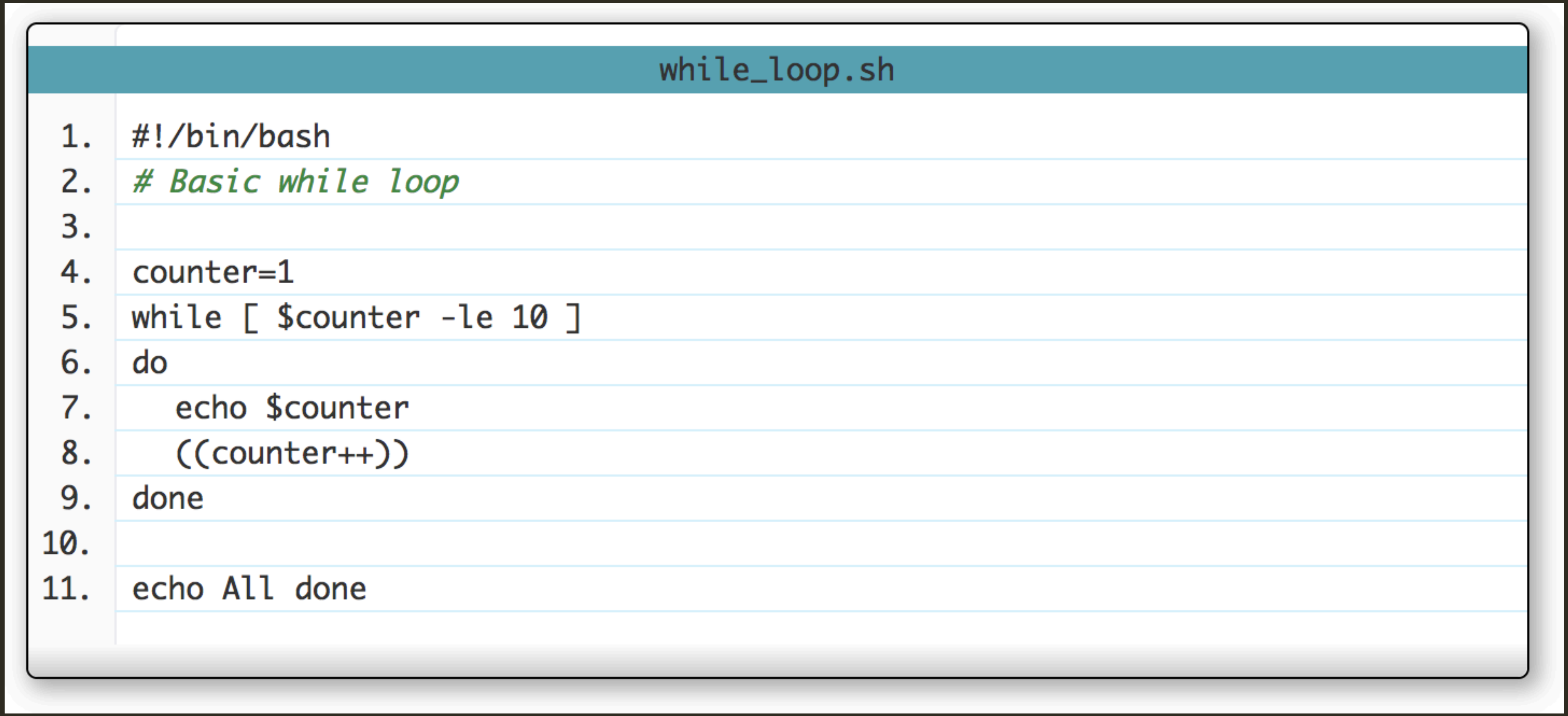
OR | |



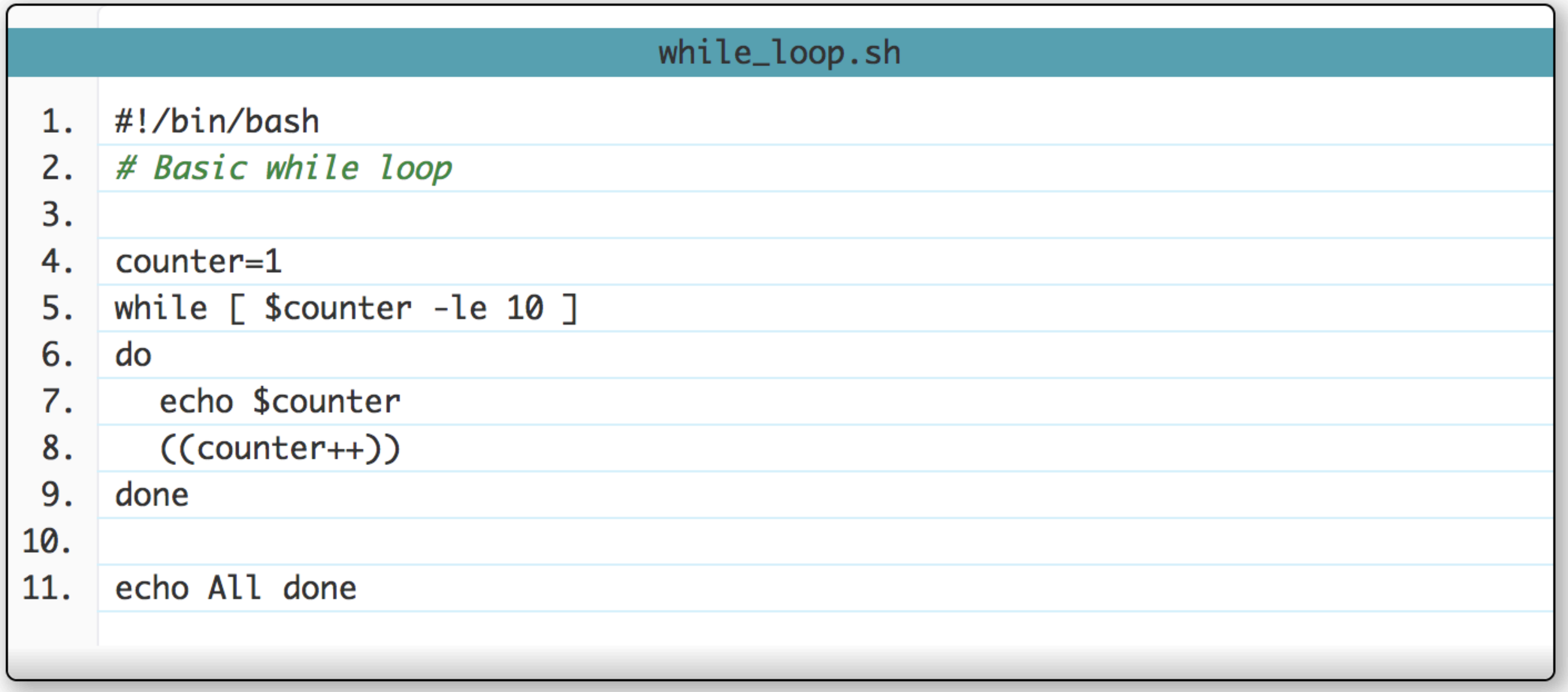
or.sh

```
#!/bin/bash
# or example

if [ $USER == 'bob' ] || [ $USER == 'andy' ]
then
  ls -alh
else
  ls
fi
```

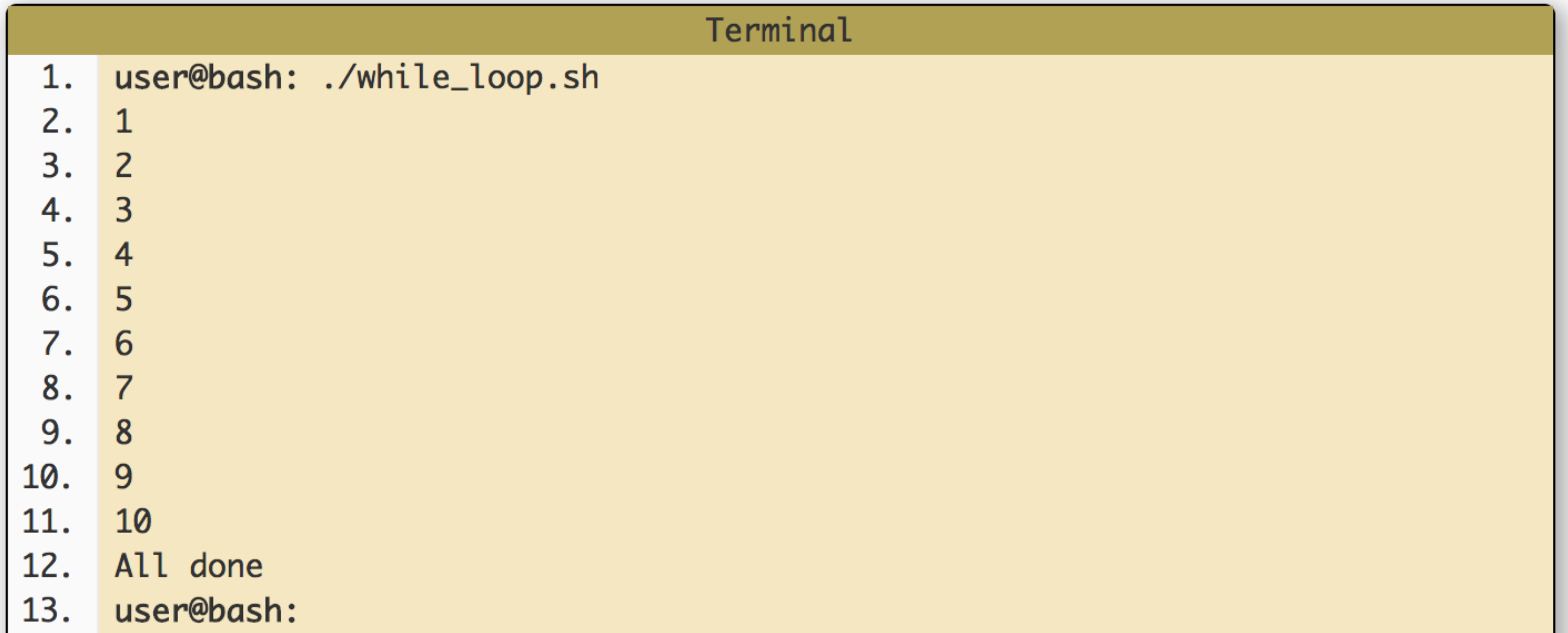
WHILE LOOPS – FOR COUNTING WITH ARITHMETIC


while_loop.sh

```
#!/bin/bash
# Basic while loop

counter=1
while [ $counter -le 10 ]
do
   echo $counter
   ((counter++))
done

echo All done
```

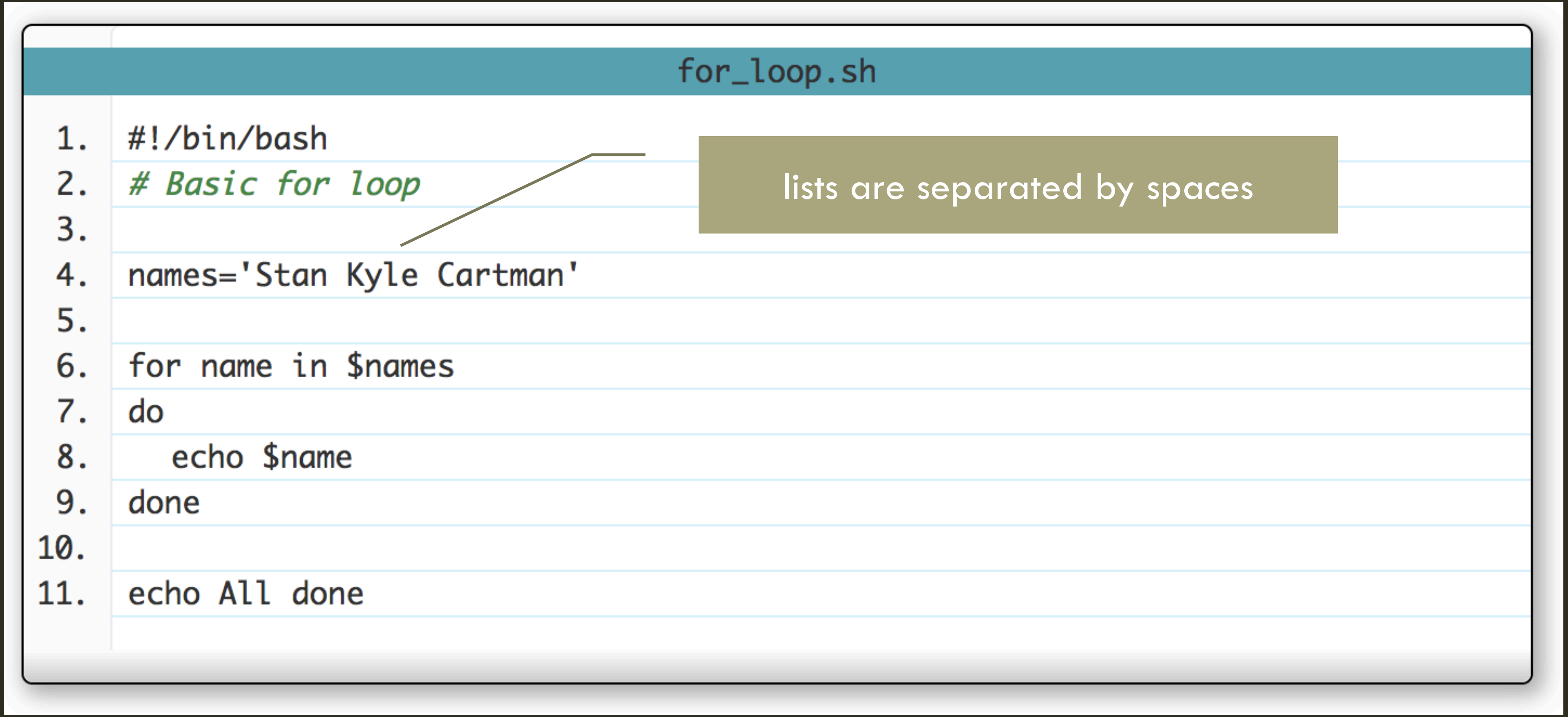

while_loop.sh

```bash
#!/bin/bash
# Basic while loop

counter=1
while [ $counter -le 10 ]
do
   echo $counter
   ((counter++))
done

echo All done
```


Terminal

```
user@bash: ./while_loop.sh
1
2
3
4
5
6
7
8
9
10
All done
user@bash:
```

FOR LOOPS – FOR LOOPING OVER LISTS



for_loop.sh

```
#!/bin/bash
# Basic for loop

names='Stan Kyle Cartman'

for name in $names
do
   echo $name
done

echo All done
```

lists are separated by spaces

for_loop.sh

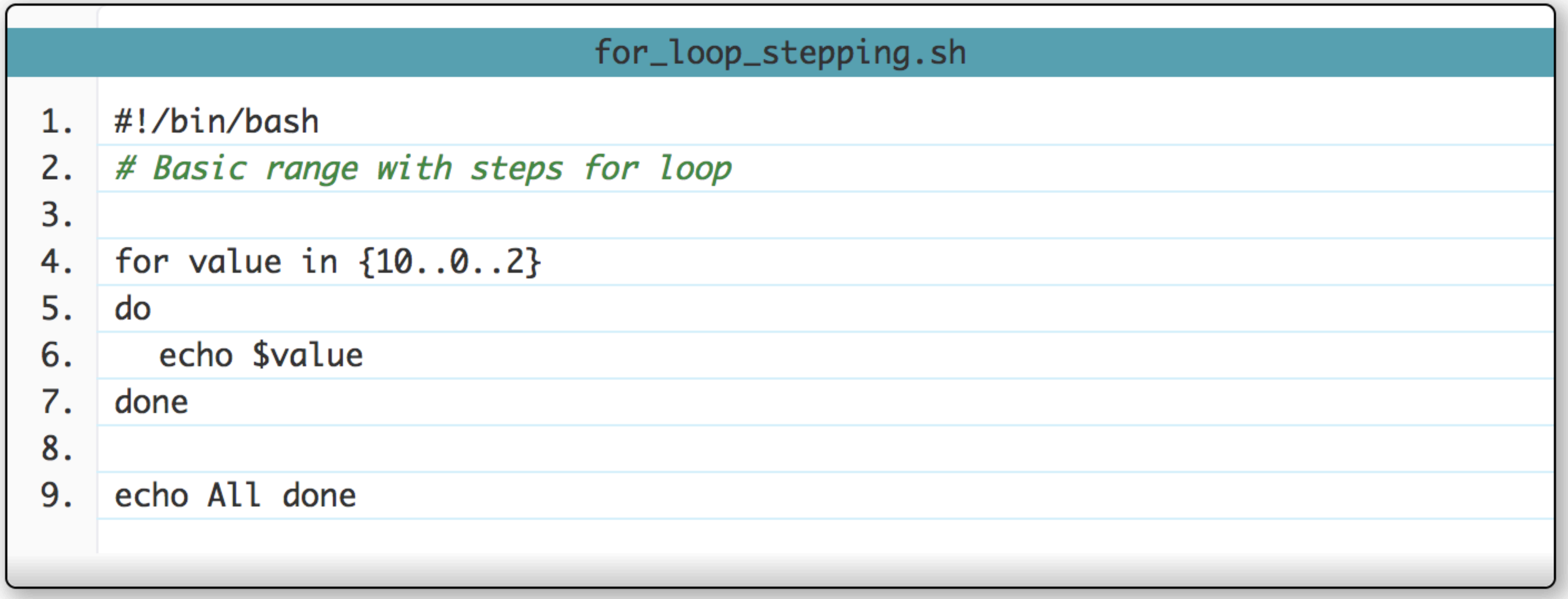
```bash
#!/bin/bash
# Basic for loop

names='Stan Kyle Cartman'

for name in $names
do
   echo $name
done

echo All done
```

Terminal

```
user@bash: ./for_loop.sh
Stan
Kyle
Cartman
All done
user@bash:
```

OTHER FOR LOOP CONTROLS

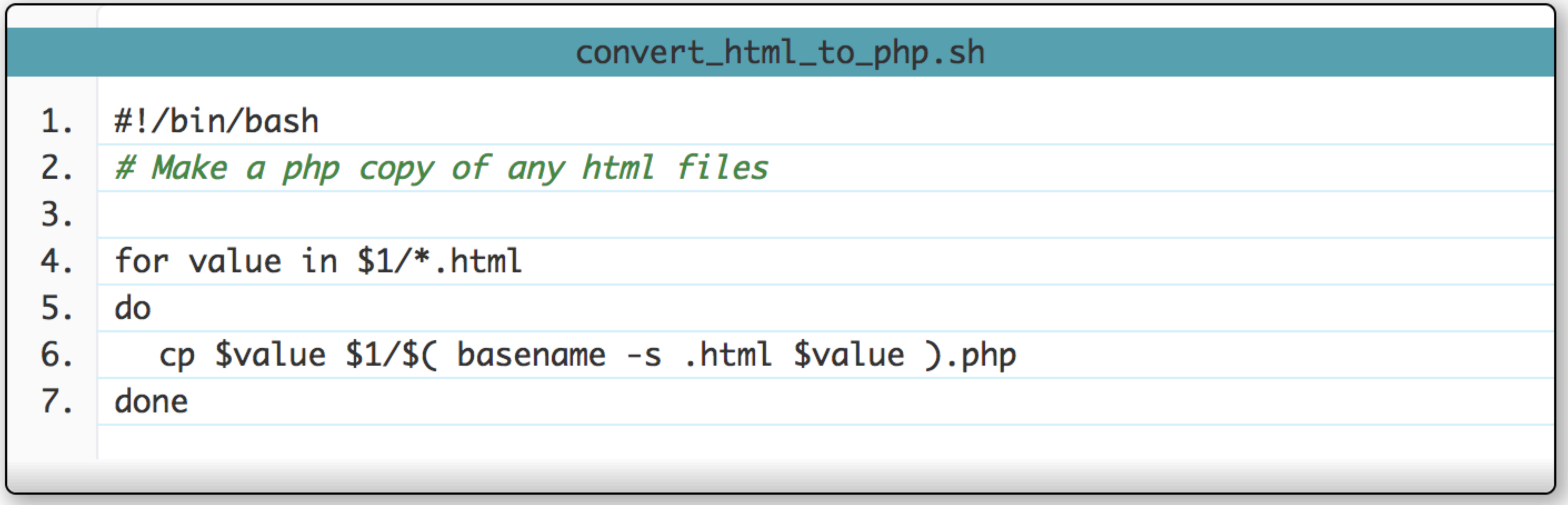
RANGES



for_loop_stepping.sh

```
#!/bin/bash
# Basic range with steps for loop

for value in {10..0..2}
do
   echo $value
done

echo All done
```

FILE LISTINGS



convert_html_to_php.sh

```
#!/bin/bash
# Make a php copy of any html files

for value in $1/*.html
do
   cp $value $1/$( basename -s .html $value ).php
done
```

SUMMARY

Getting user inputs:

- Command line arguments ($1, $2, $3, etc.)
- Read input during script execution (read varname1)
- Accept data that has been redirected into the Bash script via STDIN

Control statements:

if do elseif done – decision based on condition

while do done – loop as long as condition is true

for do done – loop over a specified list or range of integers