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

#variables
name="John"
echo $name
echo "$name"
echo "${name}!"
printf "Your name is: $name\n"
printf "Your name is also: %s\n" "$name"
echo '$name'

#math
echo $((1+2))
$((a + 200)) # add 200 to $a
echo $((RANDOM%=200)) #return a random number from 0..200

#shell exec
echo "Current Folder: $(pwd)" #posix
echo "Current Folder: `pwd`" #bashism

#redirection

cat foo.txt > bar.txt #overwrite
cat foo.txt >> bar.txt #append
cat foo.txt 2> err.log #stderr to err.log
cat foo.txt 1> out.log #stdout to out.log
cat foo.txt 2>&1 #stderr to stdout
cat foo.txt 2>&1 > out.log #stderr & stdout to logfile
cat foo.txt 2> /dev/null #mute stderr (send to dev null)
cat foo.txt &> /dev/null #mute stderr & stdout
cat foo.txt < bar.txt # send bar.txt to stdin

#PIPE!
cat foo.txt | grep bar # pipe stdout of cat to stdin of grep


#conditional exection
echo "this"; echo "then that" #both run, no matter what
echo "this" && echo "and that" #second runs if first returns 0
echo "this" || echo "or that" #second runs if first returns not 0

# if then else
if [[ -z "$name" ]]; then
   echo '$name var is empty'
elif [[ -n "$name" ]]; then
   echo "\$name var contains = \"$name\""
```

```
fi

#case statement
case "$1" in
   start | up)
       echo "starting..."
       ;;
   end | down | stop)
       echo "stopping..."
       ;;
   reboot)
       echo "rebooting..."
       ;;
   *)
       echo "Unrecognized command; beginning self-destruct..."
       echo "Usage: $0 {start|up|end|down|stop|reboot}"
       ;;
esac

#function
print_name() {
   echo "Bender Bending Rodregiuz"
}

function echo_name {
   echo "Fry"
}
echo "Name: $(print_name)"
echo "Name: $(echo_name)"

function arguments {
   echo $# # number of arguments
   echo $* # all arguments, separated by $IFS variable
   echo $@ # all arguments, separated by space
   echo $? # return code
   echo $$ # pid of command

   echo $0 # name of command
   echo $1 # first argument
   echo $2 # second argument
   echo $3 # etc...
}

#braces
echo {A,B}.txt #no spaces
echo {A,B} # echo "A B"
```

```
ls {A,B}.js #ls A.js B.js
echo {1..5} # echo "1 2 3 4 5"

echo ${name}
echo ${name/J/j} # substitution "john"
echo ${name:0:2} # slicing "Jo"
echo ${name::2}  # "Jo"
echo ${name::-1} # "Joh"
echo ${name:(-1)} # slice from end "n"
echo ${name:(-2):1} # "h"

#default values
food="banana"
echo ${food:-Cake} # $food or "Cake"
unset food
echo ${food:-"The cake is a lie"} # $food or ...
echo ${taco:=42} #set value of $taco
echo ${taco:+42} # return 42 if $taco is set
echo ${taco:?message} # show error message if $taco is not set

length=2
echo ${name:0:$length} #slice "Jo"
echo ${name:0:length}  # this too, fucking bashism

file="/path/to/foo/bar/foo.cpp"
echo ${file%.cpp} # "/path/to/foo/bar/foo"
echo ${file%.cpp}.o  # "/path/to/foo/bar/foo.o"
echo ${file%foo*} # "/path/to/foo/bar/"
echo ${file%%foo*} # "/path/to/"
echo ${file#*/} # "path/to/foo/bar/foo.cpp"
echo ${file##*/} # "foo.cpp"
echo ${file/foo/taco} #replace first foo "/path/to/taco/bar/foo.cpp"
echo ${file//foo/taco} #replace all foo "/path/to/taco/bar/taco.cpp"

echo ${file##*.} # ".cpp"
file_name=${file##*/} # "foo.cpp"
file_path=${file%$file_name} # "/path/to/foo/bar/"

echo ${#file} # return length of variable "24"

scream="HELLO"
whisper="hello"
echo ${scream,} #lowercase first char
echo ${scream,,} #lowercase all chars
echo ${whisper^} # Hello
echo ${whisper^^} # HELLO
```

```
#loops
for file in /usr/bin/*; do
   echo "${file}"
done

for ((i=0;i<100;i++)); do #c-style
   printf "$i "
done

for i in {1..5}; do #range
   echo $i
done

for i in {5..50..5}; do #step size
   echo $i
done


# test conditionals

#Strings
[[ -z $string ]] # is empty
[[ -n $string ]] # not empty
[[ $str == $string ]]
[[ $str != $string ]]

#numbers        # C equivilant
[[ $a -eq $b ]] # ==
[[ $a -ne $b ]] # !=
[[ $a -lt $b ]] # <
[[ $a -le $b ]] # <=
[[ $a -gt $b ]] # >
[[ $a -ge $b ]] # >=

#binary
[[ ! EXPR ]] # NOT
[[ X ]] && [[ Y ]] # AND
[[ X ]] || [[ Y ]] # OR

#file
[[ -e $file ]] # exists
[[ -r $file ]] # readable
[[ -h $file ]] # symlink
[[ -d $file ]] # directory
[[ -w $file ]] # writeable
```

```
[[ -s $file ]] # size > 0 bytes
[[ -f $file ]] # regular file
[[ -x $file ]] # is executable
[[ $a -nt $b ]] # newer than
[[ $a -ot $b ]] # older than
[[ $a -ef $b ]] # equal files

# arrays

fruit=('apple', 'grape', 'orange')

fruits[0]="apple"
fruits[1]="grape"
fruits[2]="orange"

Fruits=("${Fruits[@]}" "Watermelon")    # Push
Fruits+=('Watermelon')                  # Also Push
Fruits=( ${Fruits[@]/Ap*/} )            # Remove by regex match
unset Fruits[2]                         # Remove one item
Fruits=("${Fruits[@]}")                 # Duplicate
Fruits=("${Fruits[@]}" "${Veggies[@]}") # Concatenate
lines=(`cat "logfile"`)                 # Read from file


#iteration
for i in "${arrayName[@]}"; do
 echo $i
done

#brackets & arrays
echo ${Fruits[0]}           # Element #0
echo ${Fruits[@]}           # All elements, space-separated
echo ${#Fruits[@]}          # Number of elements
echo ${#Fruits}             # String length of the 1st element
echo ${#Fruits[3]}          # String length of the Nth element
echo ${Fruits[@]:3:2}       # Range (from position 3, length 2)

#dictionaries (hash table)
declare -A sounds

sounds[dog]="bark"
sounds[cow]="moo"
sounds[bird]="tweet"
sounds[wolf]="howl"

echo ${sounds[dog]} # Dog's sound
```

```
echo ${sounds[@]}   # All values
echo ${!sounds[@]}  # All keys
echo ${#sounds[@]}  # Number of elements
unset sounds[dog]   # Delete dog

#iterate over value
for val in "${sounds[@]}"; do
 echo $val
done

#iterate over key
for key in "${!sounds[@]}"; do
 echo $key
done

#parse command line options
while [[ "$1" =~ ^- && ! "$1" == "--" ]]; do case $1 in
 -V | --version )
   echo $version
   exit
   ;;
 -s | --string )
   shift; string=$1
   ;;
 -f | --flag )
   flag=1
   ;;
esac; shift; done
if [[ "$1" == '--' ]]; then shift; fi

#get user input
echo -n "Would you like to play a game? [yes/no]: "
read ans
echo $ans

echo -n "How about a nice game of global thermo nuclear war? [yes/no]: "
read -n 1 ans # just one char (no enter)
echo $ans
```