Part 2: Symmetric Cryptography

Symmetric encryption:

```
openssl enc -e -aes-128-cbc -pbkdf2 -kfile key_file.pem -in plaintext.txt -out ciphertext.txt -base64
```

Symmetric decryption:

```
openssl enc -d -aes-128-cbc -pbkdf2 -kfile key_file.pem -in ciphertext.txt -out plaintext.dec -base64
```

Generate a random symmetric key:

```
openssl rand -base64 128
```

Part 3: Asymmetric (Public Key) Cryptography

Generate private key:

```
openssl genpkey -algorithm RSA -out private.pem
```

Generate public key from private key:

```
openssl pkey -in private.pem -pubout -out public.pem
```

Encrypt with public key:

```
openssl pkeyutl -encrypt -in plaintext.txt -pubin -inkey public.pem -out ciphertext.txt
```

Decrypt with private key:

```
openssl pkeyutl -decrypt -in ciphertext.txt -inkey private.pem -out decrypted.txt
```

Part 4: Bash Basics

```
# print something to console
echo "Hello there"

# create a variable - it is important not to put a space before/after =
name="John Smith"

# use variable contents
echo "Hello there, $name"

# output the result of a command to a file instead of printing to console
echo "Hi, $name" > greetings.txt

# output contents of a file to the console
cat greetings.txt

# print file contents to console with additional info - good for debugging
echo "Contents of greetings.txt: $(cat greetings.txt)

# assign a variable to the result of a command
contents_of_greetings=$(cat greetings.txt)
echo "contents_of_greetings variable: $contents_of_greetings"
```

```
# pipe the results of one command into another
echo "message" | openssl enc -e -aes-128-cbc -k "key" -pbkdf2

# while loop - don't forget done at the end
while true
do
        # get user input, write to variable
        read -p "Please input a word: " word

        # if/then statement - don't forget fi at the end
        if [ "$word" = "cryptography" ]
        then
                echo "The word you entered was cryptography"
                # while loop exits only when the entered word is cryptography
                break
        else
                echo "The word you entered was not cryptography"
        fi
done

# create and iterate through list - no spaces before/after =, must have spaces before/after each item
item_list=( "item1" "item2" "item3" )
for item in "${item_list[@]}"
do
        echo $item
done
```

Running bash scripts

- To run a bash script, run this command in the terminal: ./bash-basics.sh
- If you get a permission denied error, run this command: chmod +x bash-basics.sh
- Use Ctrl + c to kill a bash script

To find out more about a command and any flags it might use, type man insert_command_here into the terminal, and you can exit by entering q

Netcat for Ubuntu:

```
Listening socket:     echo "message to client" | nc -l -N 1234
Connecting socket:    echo "message to server" | nc -N localhost 1234
```

Might need to modify netcat for Kali:

```
Listening socket:     echo "message to client" | nc -l -p 1234 -q 1
Connecting socket:    echo "message to server" | nc localhost 1234
```

Part 5: Hash Functions

HMAC:

```
openssl dgst -sha256 -hmac "My secret key" plaintext.txt
```