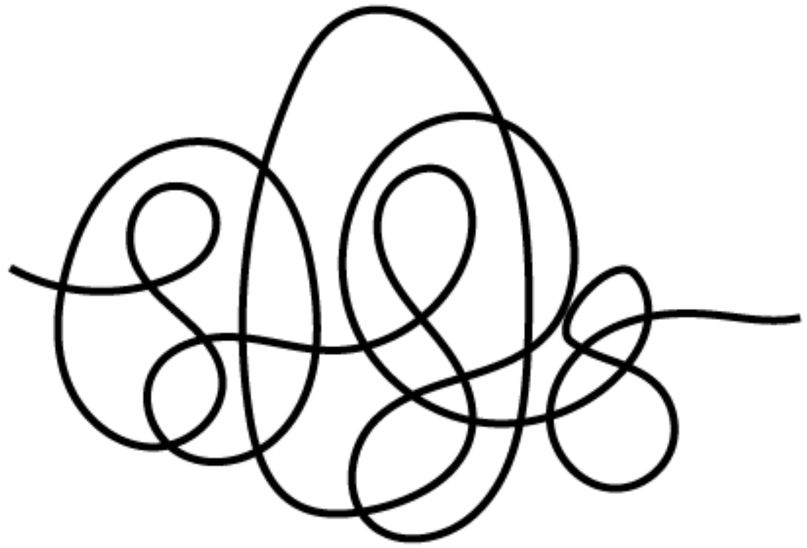
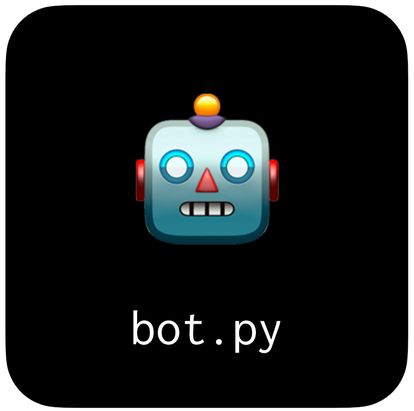


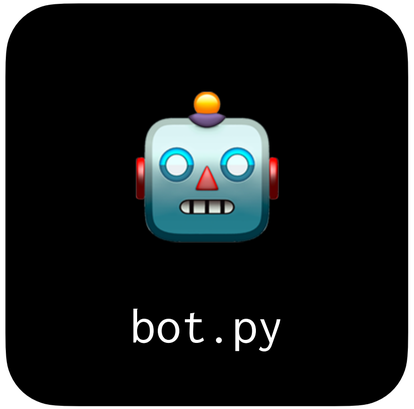
Lab 3

CS50 for MBAs

carterzenke.me/lab

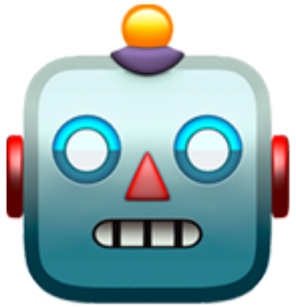

bot.py


bitcoin.py


bot.py


bitcoin.py


test_bot.py



Bot

Regular Expressions

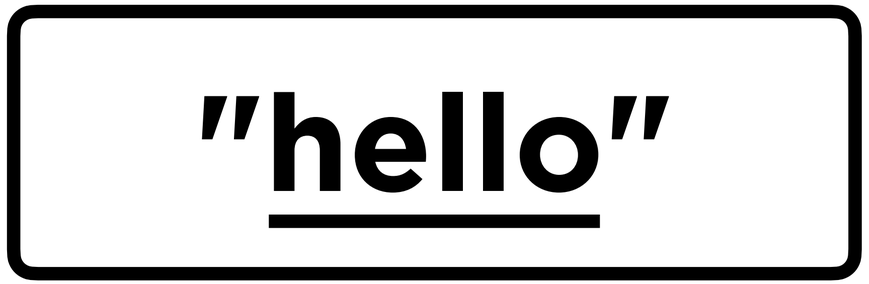
```
re.search(r"hello", text)
```

text

"hello"

```
re.search(r"hello", text)
```

text



```
re.search(r"hello", text)
```

text

"hellllloo"

Building Regular Expressions

`.`	match any character except a newline (\n)
`*`	match 0 or more repetitions
`+`	match 1 or more repetitions
`?`	match 0 or 1 repetition
`{m}`	match m repetitions
`{m,n}`	match m-n repetitions

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

```
re.search(r"hell+o", text)
```

text

"hellllo"

Building Regular Expressions

```
[]          match any character in brackets
[^]         match any character NOT in brackets
```

```
re.search(r"h[eu]llo", text)
```

text

"hullo"

```
re.search(r"h[eu]llo", text)
```

text

"hullo"

```
re.search(r"h[eu]llo", text)
```

text

"hullo"

```
re.search(r"h[eu]llo", text)
```

text

Building Regular Expressions

`()`	a group of characters
`a|b`	match a OR b

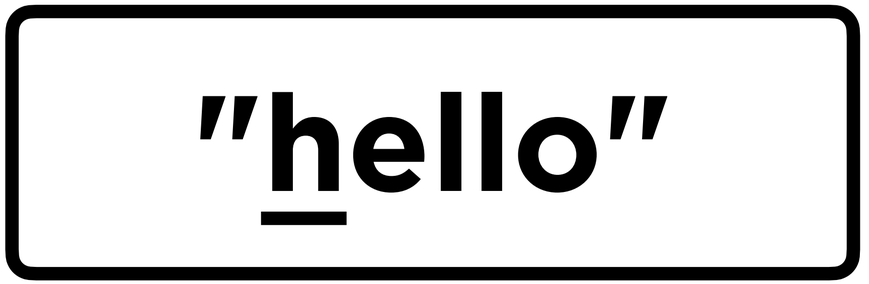
```
re.search(r"h(ello|i)", text)
```

text

"hello"

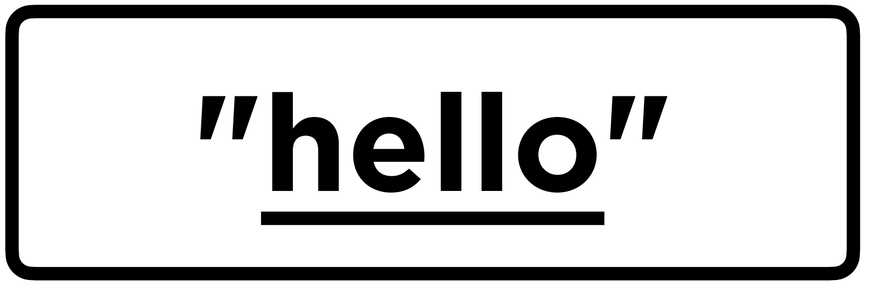
```
re.search(r"h(ello|i)", text)
```

text



```
re.search(r"h(ello|i)", text)
```

text



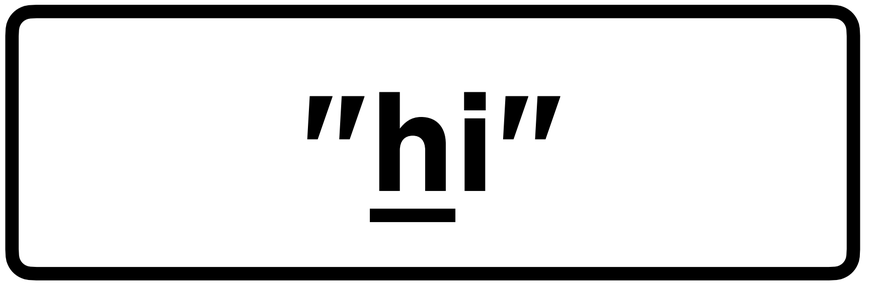
```
re.search(r"h(ello|i)", text)
```

text

"hi"

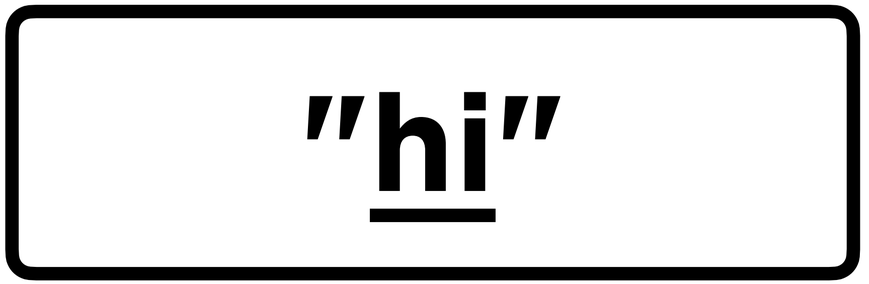
```
re.search(r"h(ello|i)", text)
```

text



```
re.search(r"h(ello|i)", text)
```

text



Capture Groups

```
match = re.search(r"I'm (.+)", text)
```

text

"I'm Carter"

```
match = re.search(r"I'm (.+)", text)
```

text

"I'm Carter"

```
match = re.search(r"I'm (.+)", text)
if matches:
    return matches.group(1)
```

text

"I'm Carter"

```
match = re.search(r"I'm (.+)", text)
if matches:
    return matches.group(1)
```

text

"I'm Carter"

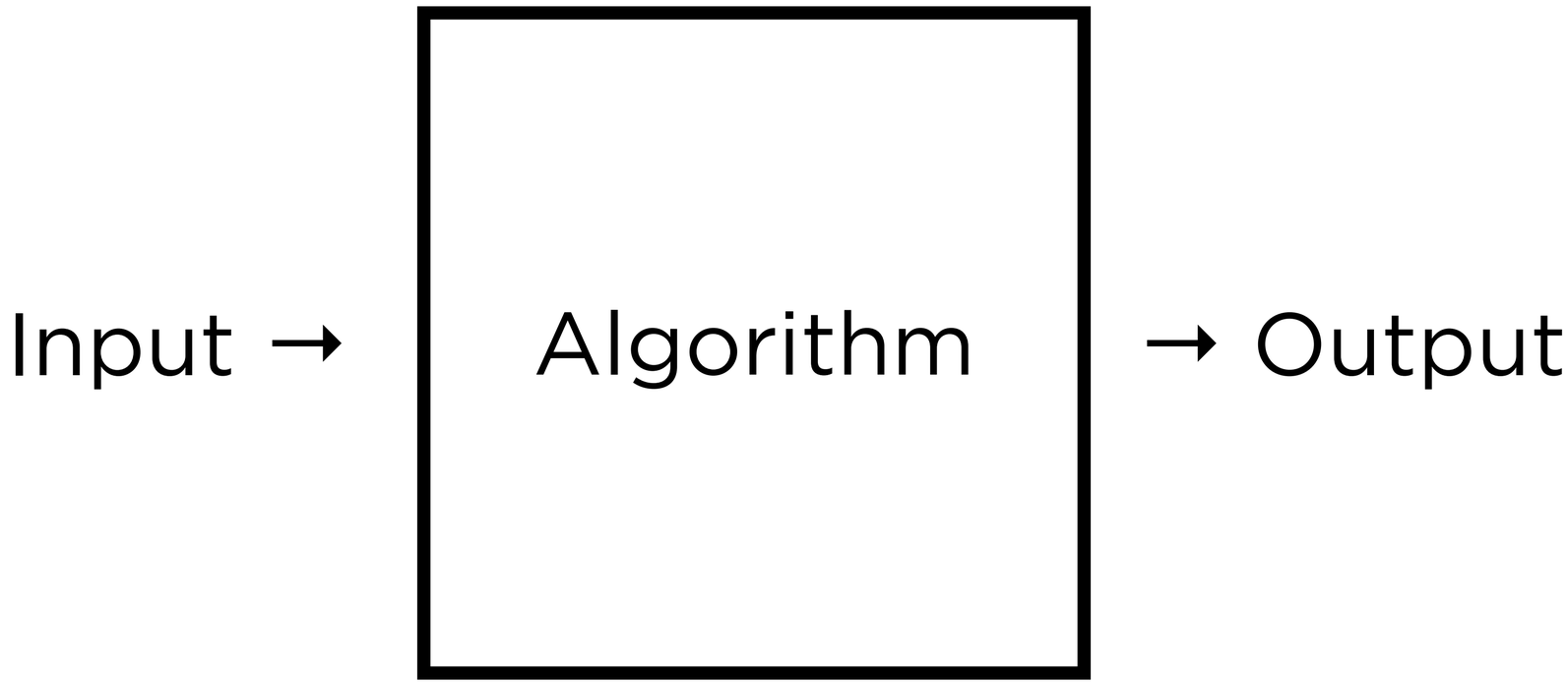
matches.group(1)

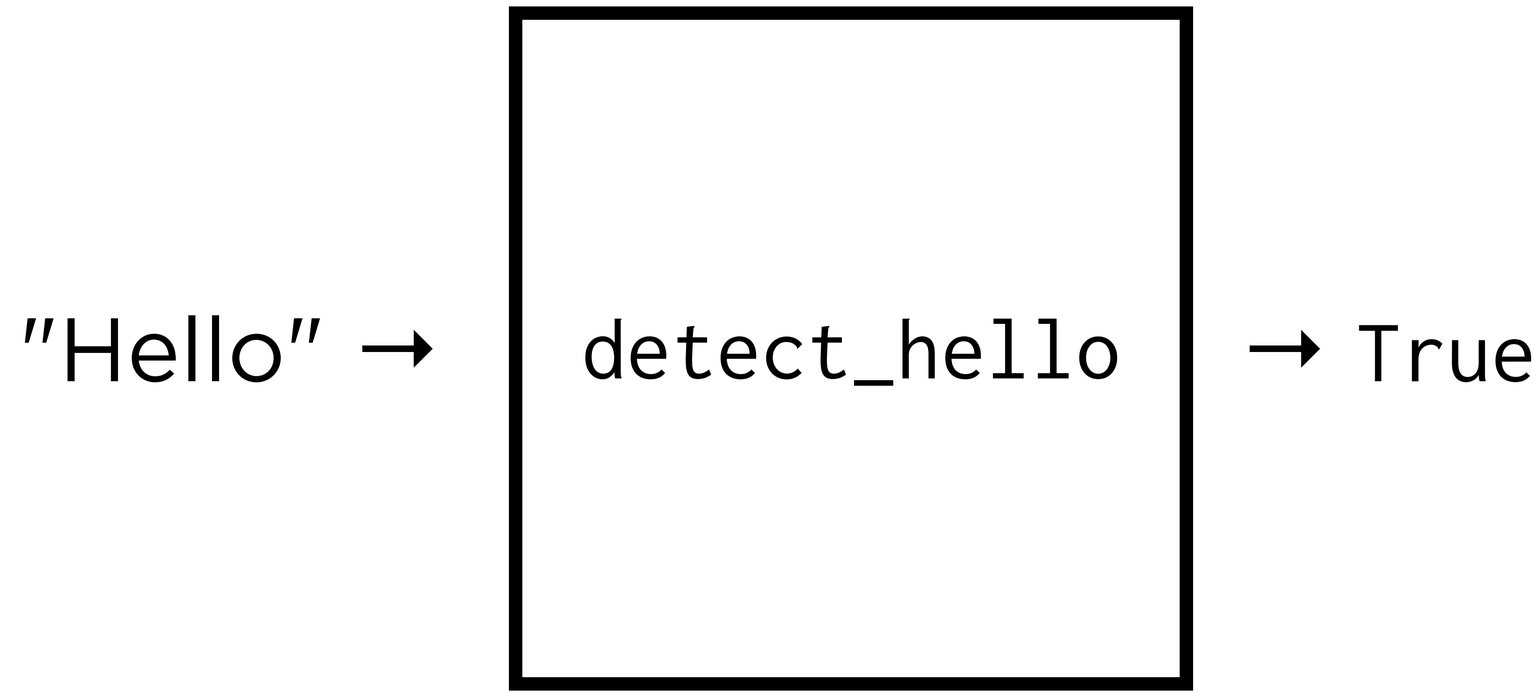
"Carter"

Name recognition

- Implement a function, **find_name**, to return the name in the following phrases:
  - "I'm NAME"
  - "My name is NAME"
  - "Name's NAME"

Unit Tests


Input →
Algorithm
→ Output


"Hello" →
detect_hello
→ True

"Hi" → detect_hello → True

"Bye" → detect_hello → False

"Hm" → detect_hello → False

"Hey" → detect_hello → True

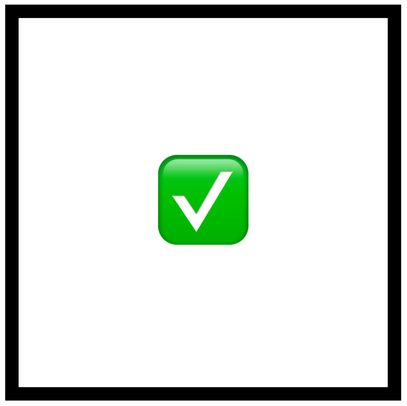
"Hello" → detect_hello → True

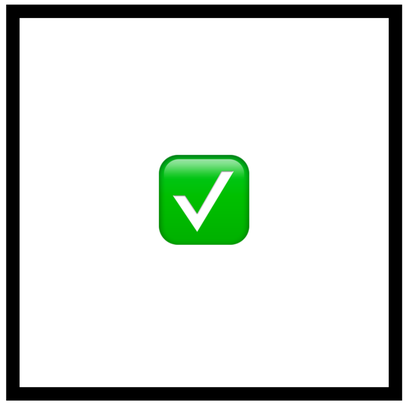
"Help!" → detect_hello → True

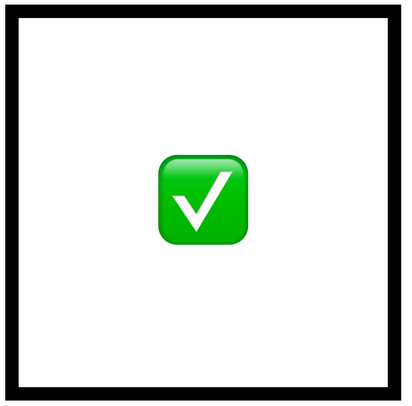
"Hola" → detect_hello → True

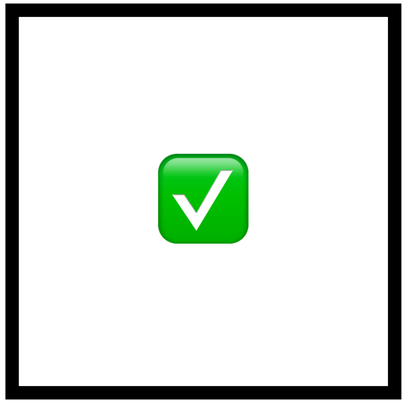
"Hiii" → detect_hello → True

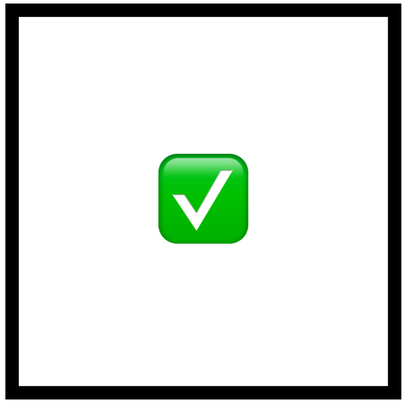
"Cya" → detect_hello → False

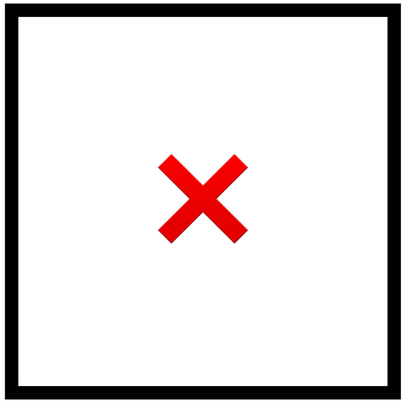
"Hi" →  → `True`

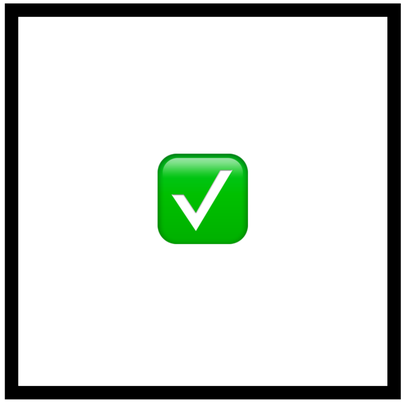
"Bye" →  → `False`

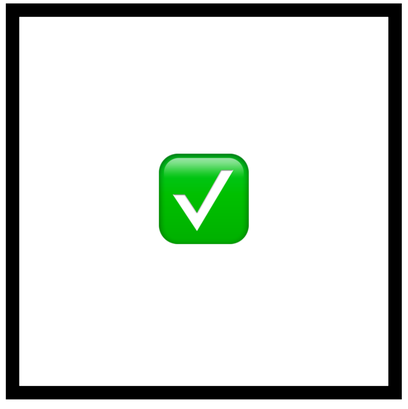
"Hm" →  → `False`

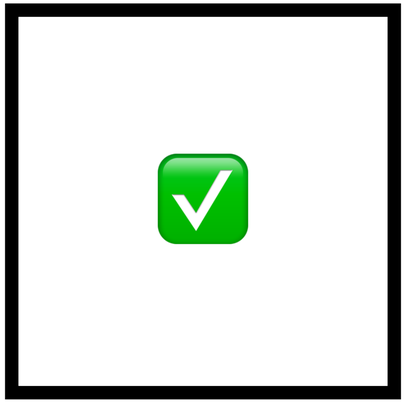
"Hey" →  → `True`

"Hello" →  → `True`

"Help!" →  → `True`

"Hola" →  → `True`

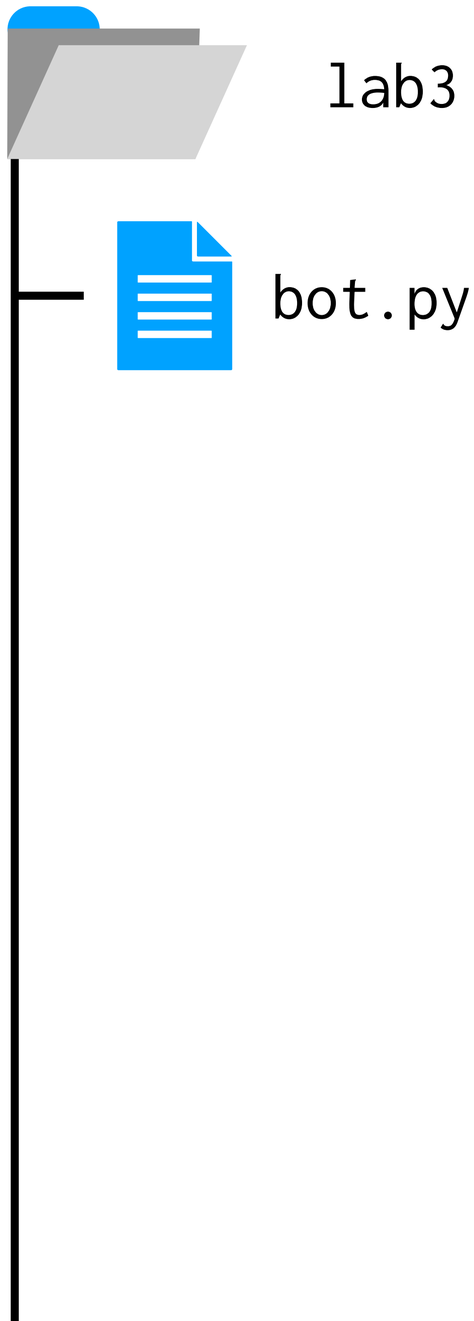
"Hiii" →  → `True`

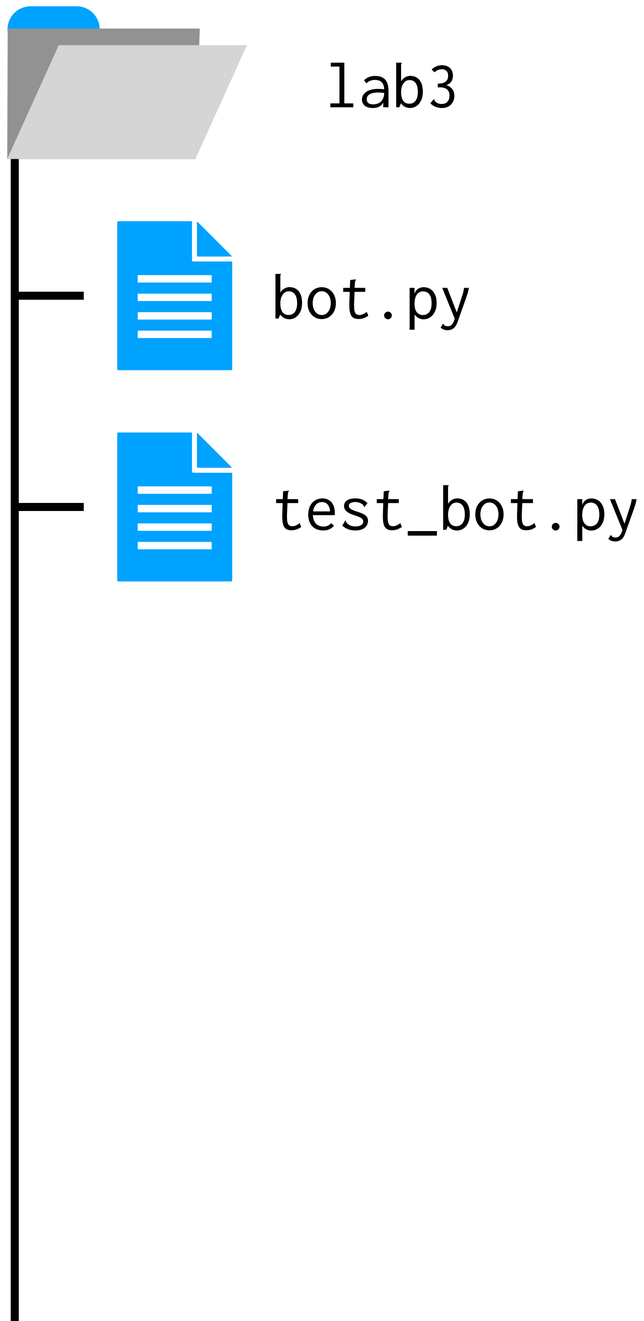
"Cya" →  → `False`

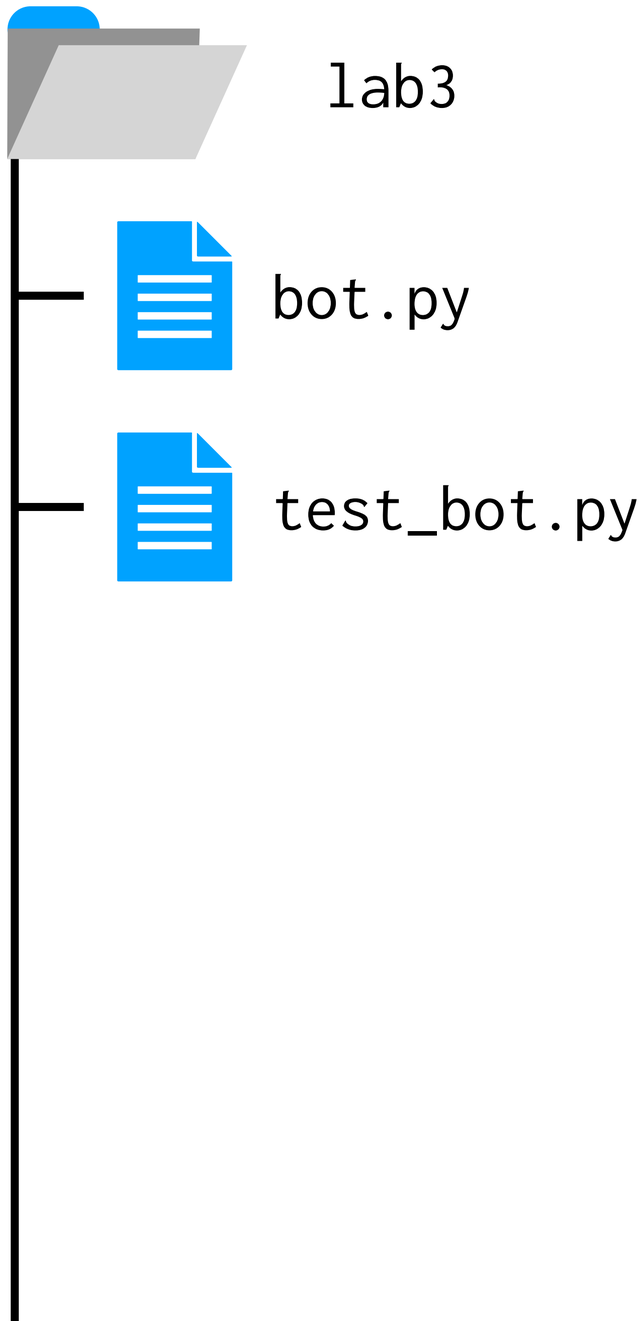
✅ Unit Tests

A program to test particular "units" of *other* programs (often single functions).

pytest


lab3
bot.py


lab3
bot.py
test_bot.py



test_bot.py

```
```

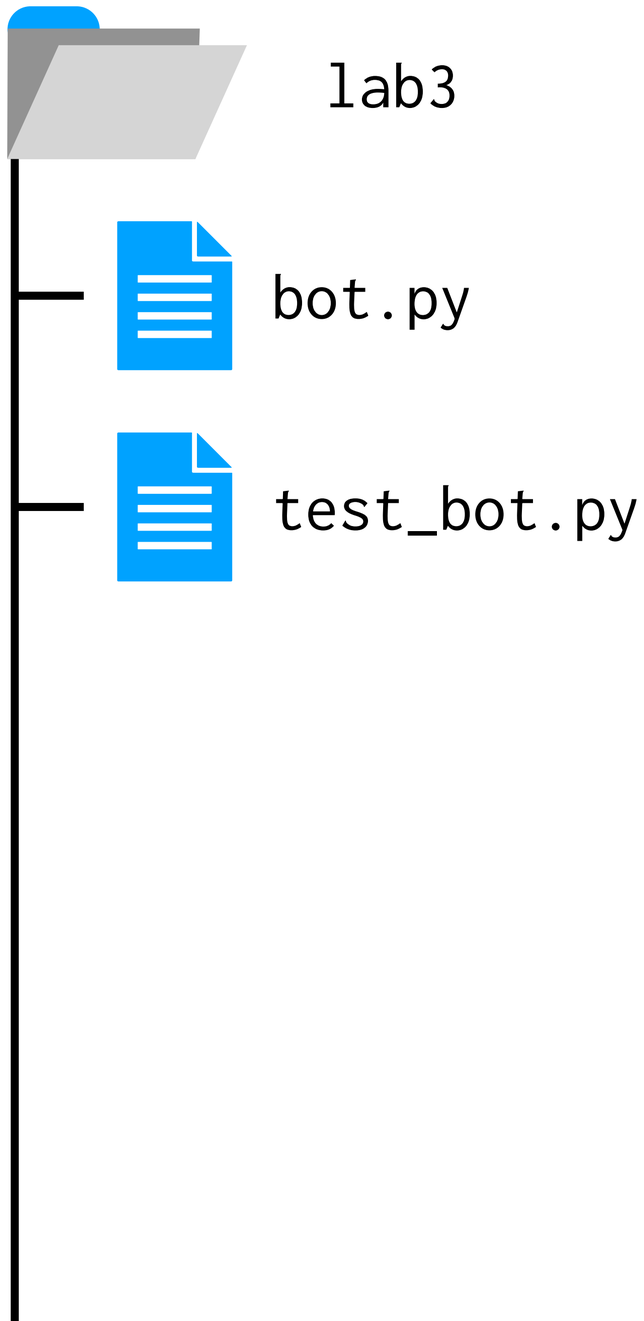
lab3

bot.py

test_bot.py

test_bot.py

```
from bot import detect_hello
```



test_bot.py

```
from bot import detect_hello

def test_detect_hello():
    ...
```

lab3

bot.py

test_bot.py

test_bot.py

```
from bot import detect_hello

def test_detect_hello():
    assert ...
```

lab3

bot.py

test_bot.py

test_bot.py

```
from bot import detect_hello

def test_detect_hello():
    assert ...
    assert ...
    assert ...
    ...
```

test_bot.py

```
from bot import detect_hello

def test_detect_hello():
    assert detect_hello("Hello") == True
    assert detect_hello("Goodbye") == False
    assert detect_hello("Helloooo") == True
    ...
```

test_bot.py

```
from bot import detect_hello

def test_detect_hello():
    assert detect_hello("Hello") == True
    assert detect_hello("Goodbye") == False
    assert detect_hello("Helloooo") == True
    ...
```

bot.py

```
...
main()
```

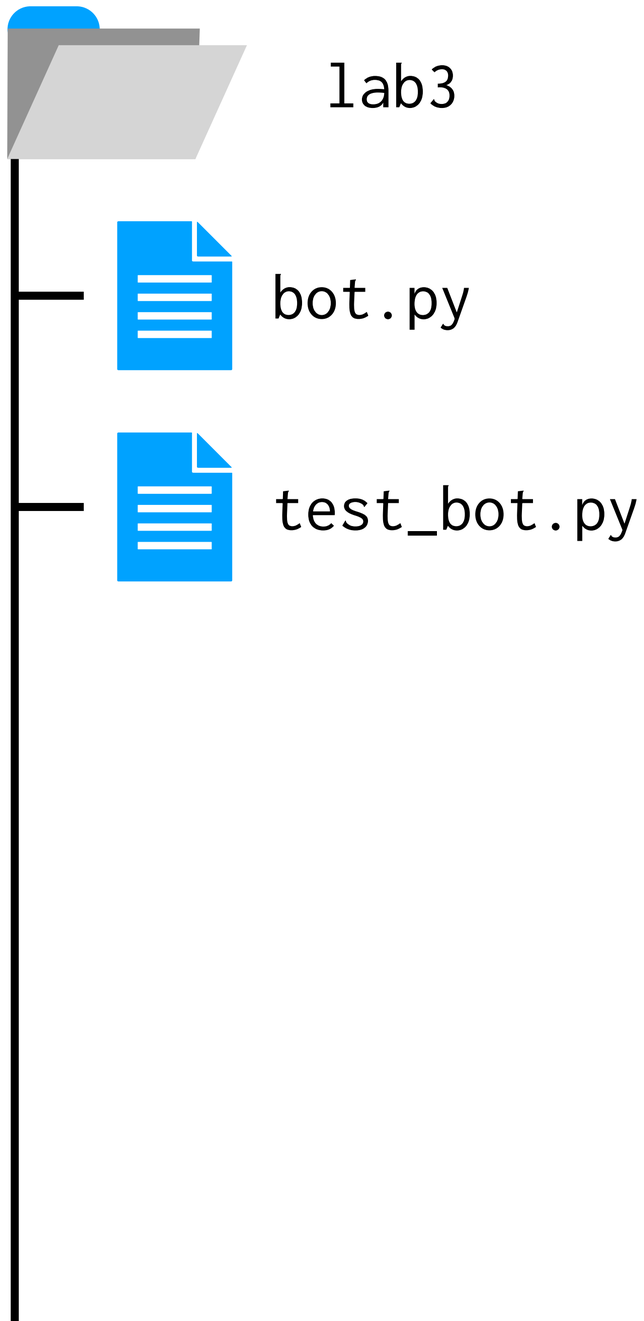
test_bot.py

```
from bot import detect_hello

def test_detect_hello():
    assert detect_hello("Hello") == True
    assert detect_hello("Goodbye") == False
    assert detect_hello("Helloooo") == True
    ...
```

bot.py

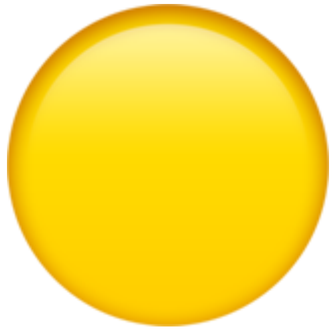
```
...
if __name__ == "__main__":
    main()
```




```
$ pytest test_bot.py
```

Your turn

- Implement additional **assert**s to thoroughly test **detect_hello**
- Add a **test_detect_goodbye** function to thoroughly test **detect_goodbye**



Bitcoin

APIs

Application Programming Interfaces

https://api.coindesk.com/v1/bpi/currentprice.json

JSON

JavaScript Object Notation

JavaScript Object Notation

A standardized way to exchange
data using key-value pairs

{"time":{"updated":"Feb 9, 2023 16:58:00 UTC","updatedISO":"2023-02-09T16:58:00+00:00","updateduk":"Feb 9, 2023 at 16:58 GMT"},"disclaimer":"This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org","chartName":"Bitcoin","bpi":{"USD":{"code":"USD","symbol":"$","rate":"22,506.8852","description":"United States Dollar","rate_float":22506.8852},"GBP":{"code":"GBP","symbol":"£","rate":"18,806.5733","description":"British Pound Sterling","rate_float":18806.5733},"EUR":{"code":"EUR","symbol":"€","rate":"21,924.9922","description":"Euro","rate_float":21924.9922}}}

```
{
  "time": {
    "updated": "Feb 9, 2023 16:58:00 UTC",
    "updatedISO": "2023-02-09T16:58:00+00:00",
    "updateduk": "Feb 9, 2023 at 16:58 GMT"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly
conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "22,506.8852",
      "description": "United States Dollar",
      "rate_float": 22506.8852
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "18,806.5733",
      "description": "British Pound Sterling",
      "rate_float": 18806.5733
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "21,924.9922",
      "description": "Euro",
      "rate_float": 21924.9922
    }
  }
}
```

```
pip install requests
```

```
{
  "time": {
    "updated": "Feb 9, 2023 16:58:00 UTC",
    "updatedISO": "2023-02-09T16:58:00+00:00",
    "updateduk": "Feb 9, 2023 at 16:58 GMT"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly
conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "22,506.8852",
      "description": "United States Dollar",
      "rate_float": 22506.8852
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "18,806.5733",
      "description": "British Pound Sterling",
      "rate_float": 18806.5733
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "21,924.9922",
      "description": "Euro",
      "rate_float": 21924.9922
    }
  }
}
```

```
{
  "time": {
    "updated": "Feb 9, 2023 16:58:00 UTC",
    "updatedISO": "2023-02-09T16:58:00+00:00",
    "updateduk": "Feb 9, 2023 at 16:58 GMT"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly
conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "22,506.8852",
      "description": "United States Dollar",
      "rate_float": 22506.8852
    },
    "GBP": {
      "code": "GBP",
      "symbol": "&pound;",
      "rate": "18,806.5733",
      "description": "British Pound Sterling",
      "rate_float": 18806.5733
    },
    "EUR": {
      "code": "EUR",
      "symbol": "&euro;",
      "rate": "21,924.9922",
      "description": "Euro",
      "rate_float": 21924.9922
    }
  }
}
```

```
{
  ...,
  ...,
  ...,
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "22,506.8852",
      "description": "United States Dollar",
      "rate_float": 22506.8852
    },
  ...
}
```

```
{
   ...,
   ...,
   ...,
   "bpi": {
     "USD": {
       "code": "USD",
       "symbol": "&#36;",
       "rate": "22,506.8852",
       "description": "United States Dollar",
       "rate_float": 22506.8852
     },
   ...
}
```

response

```
{
  ...,
  ...,
  ...,
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "&#36;",
      "rate": "22,506.8852",
      "description": "United States Dollar",
      "rate_float": 22506.8852
    },
  ...
}
```


```
response["bpi"]
```

```
{
  "USD": {
    "code": "USD",
    "symbol": "&#36;",
    "rate": "22,506.8852",
    "description": "United States Dollar",
    "rate_float": 22506.8852
  },
  ...,
}
```


```
response["bpi"]
```

```
{
  "USD": {
    "code": "USD",
    "symbol": "&#36;",
    "rate": "22,506.8852",
    "description": "United States Dollar",
    "rate_float": 22506.8852
  },
  ...,
}
```

```
response["bpi"]["USD"]
```

```
{
  "code": "USD",
  "symbol": "&#36;",
  "rate": "22,506.8852",
  "description": "United States Dollar",
  "rate_float": 22506.8852
}
```


```
response["bpi"]["USD"]
```

```
{
  "code": "USD",
  "symbol": "&#36;",
  "rate": "22,506.8852",
  "description": "United States Dollar",
  "rate_float": 22506.8852
}
```


```
response["bpi"]["USD"]["rate_float"]
```

22506.8852

```
response["bpi"]["USD"]["rate_float"]
```

Command-Line Arguments

```
$ python bitcoin.py
```

```
$ python bitcoin.py 2
```

```
$ python bitcoin.py 3.5
```

sys.argv

```
$ python bitcoin.py 3.5
```

sys.argv

['bitcoin.py', '3.5']

```
$ python bitcoin.py
```

sys.argv

['bitcoin.py']

Proper Usage

- If the user does *not* enter a number of coins along with **bitcoin.py**, print an error message and exit the program.
- Hints:
  - **sys.argv** is a list. How long should it be?

Price Index

- Print the price of X bitcoins, where X is the number of coins a user has typed as…
  - `python bitcoin.py X`
- Hints:
  - **sys.argv** is a list, with elements accessed via []
  - The type of X will at first be a **str**.

Transcribe

APIs

Application Programming Interfaces

https://cloud.google.com/speech-to-text

https://pypi.org/project/SpeechRecognition/

```
pip install SpeechRecognition
```

Submission

- **Submit code files to Gradescope** by Friday, February 10, 3:10 PM.
- Graded based on completion, but please double check to be sure your files are named correctly:
  - `bot.py` **not** `bot (1).py`