Flask

Flask

- Python is not just used for command-line programming, though that's a major use case.

- Python contains native functionality to support networking and more, enabling site backends to be written in Python.

Flask

- Web frameworks make this process much easier, abstracting away the minutia of Python's syntax and providing helper functions.

- Some of the most popular include: Django, Pyramid, and Flask.

- We use Flask in CS50 because it is lightweight for ease of use in CS50 IDE, while still being feature-rich.

Flask

- We know that we can use HTML to build websites, but websites built using pure HTML suffer from a serious limitation.

- Imagine we want to create a website that displays the current time in Cambridge, MA, displaying it to the latest minute.

Flask

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:08
    </body>
</html>
```

Flask

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:09
    </body>
</html>
```

Flask

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:10
    </body>
</html>
```

Flask

```
<html>
    <head>
        <title>Current Time in Cambridge</title>
    </head>
    <body>
        The current time in Cambridge is 14:11
    </body>
</html>
```

Flask

- Websites that are pure HTML are completely static. The only way we can update the content of our pages is to manually open up our source files, edit and save, and then the next time the user visits or refreshes the page they'll get the content.

- Incorporating Python into our code can make our code quite a bit more flexible and introduce a way for our pages to update or be dynamic without requiring our intervention.

Flask

```
from flask import Flask
from datetime import datetime
from pytz import timezone

app = Flask(__name__)

@app.route("/")
def time():
    now = datetime.now(timezone('America/New_York'))
    return "The current date and time in Cambridge is {}".format(now)
```

Flask

```
from flask import Flask
from datetime import datetime
from pytz import timezone

app = Flask(__name__)

@app.route("/")
def time():
    now = datetime.now(timezone('America/New_York'))
    return "The current date and time in Cambridge is {}".format(now)
```

Flask

```
from flask import Flask
from datetime import datetime
from pytz import timezone

app = Flask(__name__)

@app.route("/")
def time():
    now = datetime.now(timezone('America/New_York'))
    return "The current date and time in Cambridge is {}".format(now)
```

Flask

```
from flask import Flask
from datetime import datetime
from pytz import timezone

app = Flask(__name__)

@app.route("/")
def time():
    now = datetime.now(timezone('America/New_York'))
    return "The current date and time in Cambridge is {}".format(now)
```

Flask

- It’s pretty simple to get started using Flask within CS50 IDE.

Flask

- It’s pretty simple to get started using Flask within CS50 IDE.

```
from flask import Flask
```

Flask

- It’s pretty simple to get started using Flask within CS50 IDE.

- After importing the Flask module, we need to initiate a Flask application.

Flask

- It’s pretty simple to get started using Flask within CS50 IDE.

- After importing the Flask module, we need to initiate a Flask application.

```
app = Flask(__name__)
```

Flask

- It’s pretty simple to get started using Flask within CS50 IDE.
- After importing the Flask module, we need to initiate a Flask application.
- From there, it’s just a matter of writing functions that define the behavior of our application.

Flask

```
def index():
    return "You are at the index page!"


def sample():
    return "You are on the sample page!"
```

Flask

```
@app.route("/")
def index():
    return "You are at the index page!"

@app.route("/sample")
def sample():
    return "You are on the sample page!"
```

Flask

- The lines just added are known as “decorators.” They are used, in Flask, to associate a particular function with a particular URL.

- Decorators also have more general use in Python, but that goes beyond the scope of CS50.

Flask

- It’s also quite straightforward to run our Flask application within CS50 IDE.

```
export FLASK_APP=application.py
export FLASK_DEBUG=1
flask run
```

Flask

- It’s also quite straightforward to run our Flask application within CS50 IDE.

```
export FLASK_APP=application.py
export FLASK_DEBUG=1
flask run
```

Flask

- It’s also quite straightforward to run our Flask application within CS50 IDE.

```
flask run
```

Flask

- Data can be passed in via URLs, akin to using HTTP GET.

Flask

- Data can be passed in via URLs, akin to using HTTP GET.

```
@app.route("/show/<number>")
def show(number):
    return "You passed in {}".format(number)
```

Flask

- Data can be passed in via HTML forms, as with HTTP POST, but we need to indicate that Flask should respond to HTTP POST requests explicitly.

Flask

- Data can be passed in via HTML forms, as with HTTP POST, but we need to indicate that Flask should respond to HTTP POST requests explicitly.

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if not request.form.get("username")
        return apology("must provide username")
```

Flask

- Data can be passed in via HTML forms, as with HTTP POST, but we need to indicate that Flask should respond to HTTP POST requests explicitly.

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if not request.form.get("username")
        return apology("must provide username")
```

Flask

- Data can be passed in via HTML forms, as with HTTP POST, but we need to indicate that Flask should respond to HTTP POST requests explicitly.

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if not request.form.get("username")
        return apology("must provide username")
```

Flask

- We could also vary the behavior of our function depending on the type of HTTP request received:

Flask

- We could also vary the behavior of our function depending on the type of HTTP request received:

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == "POST":
        # do one thing
    else:
        # do a different thing
```

Flask

- We could also vary the behavior of our function depending on the type of HTTP request received:

```
@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == "POST":
        # do one thing
    else:
        # do a different thing
```

Flask

- Flask has a number of functions within its module that will be useful for application development.

Flask

- Flask has a number of functions within its module that will be useful for application development.

```
url_for()
```

Flask

- Flask has a number of functions within its module that will be useful for application development.

```
url_for()
redirect()
```

Flask

- Flask has a number of functions within its module that will be useful for application development.

```
url_for()
redirect()
session()
```

Flask

- Flask has a number of functions within its module that will be useful for application development.

```
url_for()
redirect()
session()
render_template()
```

Flask

- More information available at the Flask quick start guide:

`http://flask.pocoo.org/docs/0.12/quickstart/`

- More information on using Jinja can be found at:

`http://jinja.pocoo.org/`