

Let's Learn Python!

Young Coders at PyCon 2016

Meet your teachers:

Katie Cunningham
Barbara Shaurette



What we're doing today

- An overview of the basics
- Automating things with scripts
- Building a web site
- Making some games?

Let's talk to Python!

Math

Try doing some math
at the prompt:

```
>>> 1 + 2
```

```
>>> 12 - 3
```

```
>>> 9 + 5 - 15
```

```
>>> 6 * 5
```

```
>>> 10 * 5 * 3
```

```
>>> 6 / 2
```

Floats (decimals):

```
>>> 10/2
```

5.0

Integers:

```
>>> round(10/2)
```

5

Math

Comparison operators:

`==` Equal to

`!=` Not equal to

`<` Less than

`>` Greater than

`<=` Less than or equal to

`>=` Greater than
or equal to

Practice:

```
>>> 5 < 4 + 3
```

```
>>> 12 + 1 >= 12
```

```
>>> 16 * 2 == 32
```

```
>>> 16 != 16
```

```
>>> 5 >= 6
```

Strings

A few examples:

```
>>> 'Hello!'  
>>> '3 + 5'
```

What if you have a
quote in your string?:

```
>>> 'When\'s lunch?'  
>>> "When's lunch?"
```

Another way to return a string:

```
>>> print("Hello")  
>>> print("Hello"[1])
```

Try typing one without quotes:

```
>>> apple
```

Strings

Concatenation:

```
>>> "Hi" + "there!"
```

Multiplying:

```
>>> "HAHA" * 250
```

Can you use other math operators on strings?

```
>>> "Hi" - "there"
```

```
>>> "Hi" / "there"
```

```
>>> "Hi" > "there"
```

Variables

Calculate a value:

```
>>> 12 * 12  
144
```

Create a new variable,
give it a string value:

```
>>> color = "yellow"  
>>> color  
'yellow'
```

Store the value, give it a name:

```
>>> donuts = 12 * 12  
>>> donuts  
144
```

Assign new values:

```
>>> color = "red"  
>>> color = "fish"  
>>> color = 12
```

Variables

Math operations

```
>>> donuts = 12 * 12  
>>> fishes = 3  
>>> fishes + donuts
```

Get an index from a string:

```
>>> fruit = "watermelon"  
>>> print (fruit[2])  
't'
```

String operations

```
>>> color = "yellow"  
>>> day = "Monday"  
>>> color + day  
>>> color * fishes  
>>> color + day * fishes  
>>> (color + day) * fishes
```

Do some math to get the index:

```
>>> mynum = 3  
>>> print (fruit[mynum-2])  
'a'
```

Variables

Assigning values or making comparisons?

```
>>> fruit = "watermelon"
```

```
>>> 5 = 6
```

```
>>> fruit == "watermelon"
```

```
>>> 5 == 6
```

Errors

```
>>> "friend" + 5
```

```
Error
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: Can't convert 'int' object to str implicitly
```

What if we make 5 a string?

```
>>> "friend" + "5"
```

```
friend5
```

Errors

```
>>> 5 + "friend"
```

```
Error
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

How can we fix this expression?

```
>>> 5 + len("friend")
```

```
11
```

```
>>> print(5, "friend")
```

```
5 friend
```

Data types

```
>>> type("Hi!")
```

```
<class 'str'>
```

```
>>> type(27)
```

```
<class 'int'>
```

```
>>> type(15.238)
```

```
<class 'float'>
```

Is 1 equal to 1?

```
>>> 1 == 1
```

```
True
```

Is 15 less than 5?

```
>>> 15 < 5
```

```
False
```

```
>>> type(True)
```

```
<class 'bool'>
```

```
>>> type(False)
```

```
<class 'bool'>
```

Data type: Booleans

What happens when we type Boolean values in the interpreter?

```
>>> True  
>>> False  
>>> true  
>>> false  
>>> type(True)  
>>> type("True")
```

When the words ‘True’ and ‘False’ begin with *upper case* letters, Python knows to treat them like Booleans, not strings or integers.

Data type: Booleans

```
>>> 1==1 and 2==2  
True
```

```
>>> 1==1 and 2==3  
False
```

```
>>> 1==2 and 2==3  
False
```

```
>>> 1==1  
True
```

```
>>> not 1==1  
False
```

```
>>> not True  
False
```

```
>>> 1==1 or 2==2  
True
```

```
>>> 1==1 or 2!=2  
True
```

```
>>> 1==2 or 2==3  
False
```

Data type: list

```
>>> fruit = ["apple", "banana", "grape"]  
>>> numbers = [3, 17, -4, 8.8, 1]  
>>> mylist = [1, "blue", 1.8]  
>>> otherlist = [fruit, numbers, mylist]
```

```
>>> type(mylist)  
<class 'list'>
```

```
>>> fruit[0]  
'apple'
```

```
[ 'apple' , 'banana' , 'grape' ]  
0 1 2
```

Data type: list

```
>>> fruit = ["apple", "banana", "grape"]
>>> fruit.append("mango")
>>> fruit
['apple', 'banana', 'grape', 'mango']

>>> fruit.insert(2, "orange")
>>> fruit
['apple', 'banana', 'orange', 'grape',
'mango']
```

Data type: dict

```
>>> mydict = { 'mykey1': 'apple', 'mykey2': 3, 'mykey3': mylist}

>>> type(mydict)
<class 'dict'>

>>> mydict['mykey1']
'apple'

>>> mydict.keys()
['mykey3', 'mykey2', 'mykey1']

>>> type(mydict.keys())
<class 'list'>
```

Logic: if Statements

If a condition is met, perform an action:

```
>>> name = "Katie"  
>>> if name == "Katie":  
    print("Hi Katie!")
```

Hi Katie!

Add a choice with the *else clause*:

```
>>> if name == "Katie":  
    print("Hi Katie!")  
else:  
    print("Impostor!")
```

if Statements

Add more choices with the `elif clause`:

```
>>> if name == "Katie":  
        print("Hi Katie!")  
    elif name == "Barbara":  
        print("Hi Barbara!")  
    else:  
        person = input("Who are you? ")  
        print('Hello', person)
```

if Statements: Practice

Write an `if` statement that prints the word “Yay!” if the variable `mycolor` is "yellow".

Add an `elif clause` and an `else clause` to print two different messages for other values of `mycolor`.

if Statements: Practice

Write an `if` statement that prints the word “Yay!” if the variable `mycolor` is "yellow".

Add an `elif clause` and an `else clause` to print two different messages for other values of `mycolor`.

```
>>> color = "blue"  
>>> if color == "yellow":  
    print("Yay!")  
elif color == "purple":  
    print("Try again!")  
else:  
    print("We want yellow!")
```

Loops

Loops are chunks of code
that repeat a task over and over again.

★ *Counting loops* repeat a
certain number of times.

★ *Conditional loops* keep
going until a certain thing happens
(or as long as some condition is True).



Loops

Counting loops repeat a certain number of times - they keep going until they get to the end of a count.

```
>>> for mynum in [1, 2, 3, 4, 5]:  
    print("Hello", mynum)
```

```
Hello 1  
Hello 2  
Hello 3  
Hello 4  
Hello 5
```

The *for* keyword is used to create this kind of loop, so it is usually just called a *for loop*.

Loops

Conditional loops repeat until something happens (or as long as some condition is True).

```
>>> count = 0  
>>> while (count < 4):  
    print("The count is:", count)  
    count = count + 1
```

```
The count is: 0  
The count is: 1  
The count is: 2  
The count is: 3
```

The *while* keyword is used to create this kind of loop, so it is usually just called a *while loop*.

Functions

Define a simple function:

```
>>> def say_hello():
    print("Hello")
```

Call the function:

```
>>> say_hello()
Hello
```

Define a function with arguments:

```
>>> def say_hello(myname):
    print("Hello", myname)
```

Call the function:

```
>>> say_hello("Katie")
Hello Katie
```

```
>>> say_hello("Barbara")
Hello Barbara
```

Functions: Practice

Create a function that takes **two numbers**, multiplies them together, and prints out the result.

Functions: Practice

Create a function that takes **two numbers**, multiplies them together, and prints out the result.

```
>>> def multiply(num1, num2):  
        print(num1 * num2)
```

```
>>> multiply(4, 5)  
20
```

```
>>> multiply("hello", 5)  
hellohelloseallohelloseallo
```

Functions: Output

```
>>> def dub(mynum):  
        print(mynum*2)
```

```
>>> newnum = double(12)  
24
```

```
>>> newnum
```

```
>>> def dub(mynum):  
        return mynum*2
```

```
>>> newnum = double(12)
```

```
>>> newnum  
24
```

Functions: Keyword Arguments

```
def repeat(text, times):  
    return text * times
```

This works:

```
>>> repeat("ha", 6)  
"hahahahaha"
```

This also works:

```
>>> repeat(text="ha", times=6)  
"hahahahaha"
```

Functions

- ★ Functions are **defined** using `def`.
- ★ Functions are **called** using **parentheses**.
- ★ Functions take **arguments** and can return **outputs**.
- ★ `print` *displays* information, but does not give a value
- ★ `return` gives a **value** that can be reused

Modules

Lots of modules are included in the Python Standard Library.

Here's how you can use a few of these modules:

Generate a random number between 1-100:

```
>>> import random  
>>> random.randint(1,100)
```

What timezone does your computer think it's in?:

```
>>> import time  
>>> time.tzname
```

Print a calendar for this month!:

```
>>> import calendar  
>>> calendar.pmonth(2016, 6)
```

Modules

Print the names of all the files in a directory:

```
>>> import os  
>>> for file in os.listdir("/home/pi"):  
    print(file)
```

Open a web page and read it:

```
>>> import urllib.request  
>>> myurl = "http://www.google.com"  
>>> data = urllib.request.urlopen(myurl).read()  
>>> print(data)
```

Modules

Turtles!

```
>>> import turtle  
>>> turtle.reset()  
>>> turtle.forward(20)  
>>> turtle.right(20)  
>>> turtle.forward(20)  
>>> turtle.bye()
```

You can find out about other modules at: <http://docs.python.org>

Objects

Objects

In the real world,
objects have:

- things that you can do to them (actions)
- words that describe them (properties)

In Python:

- “things you can do” to an object are called *methods*
- “words that describe” an object are called *attributes*

Objects

Everything in Python is an object:

```
>>> name = "Andrew"
```

```
>>> name.upper()
```

```
"ANDREW"
```

```
>>> fruit = ["apple", "banana"]
```

```
>>> fruit.append("orange")
```

```
["apple", "banana", "orange"]
```

Objects

If this ball is an *object* named myBall, it might have these *attributes*:

```
myBall.color  
myBall.size  
myBall.weight
```

You can display them:

```
print myBall.size
```

You can assign values to them:

```
myBall.color = 'green'
```

You can assign them to attributes in other objects:

```
anotherBall.color = myBall.color
```



Objects

The ball object might have these *methods*:

myBall.kick()

myBall.throw()

myBall.inflate()

Methods are the things you can do
with an object.

Methods are chunks of code - *functions* -
that are included inside the object.



Objects

In Python, a *class* is like a description - or blueprint - of an object.

```
class Ball:  
  
    color = 'red'  
    size = 'small'  
    direction = 'down'  
  
    def bounce(self):  
        if self.direction == 'down':  
            self.direction = 'up'  
        else:  
            self.direction = 'down'
```



Objects

Once we create an object by defining its class, we can call it in a program by creating an instance of it.

Creating multiple instances of an object:

```
>>> my_ball = Ball()  
>>> your_ball = Ball()  
>>> her_ball = Ball()
```

Giving these instances some attributes:

```
>>> my_ball.color = "purple"  
>>> your_ball.color = "red"  
>>> her_ball.color = "yellow"
```

Now let's try out one of the methods:

```
>>> my_ball.bounce()
```

Let's write some scripts

Script: Math Homework

homework.py

```
input_file = 'math_problems.txt'

def test_problems():
    f = open(input_file)
    lines = f.read().splitlines()
    f.close()

    for line in lines:
        answer = eval(line)
        print str(line) + " = " + str(answer)

def main():
    test_problems()

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

Script: Math Homework

```
input_file = 'math_problems.txt'

def test_problems():
    f = open(input_file)
    lines = f.read().splitlines()
    f.close()

    for line in lines:
        answer = eval(line)
        print str(line) + " = " + str(answer)

def main():
    test_problems()

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

Script:Urls

Ned's List of Programming Language URLs: <http://memweb.newsguy.com/~nedbush/proglang.htm>

urls.py

```
import urllib.request

input_file = 'urls.txt'
valid_output_file = 'valid_urls.txt'

def test_urls():

    f = open(input_file)
    lines = f.read().splitlines()
    f.close()

    v = open(valid_output_file, "w")

    for line in lines:
        try:
            urllib.request.urlopen(line)
            v.write(line)
        except urllib.error.URLError as error:
            print(line, error)
        except urllib.error.HTTPError as error:
            print(line, error.code)

    v.close

def main():
    ...
```

```
import urllib.request

input_file = 'urls.txt'
valid_output_file = 'valid_urls.txt'

def test_urls():

    f = open(input_file)
    lines = f.read().splitlines()
    f.close()

    v = open(valid_output_file, "w")

    for line in lines:
        try:
            urllib.request.urlopen(line)
            v.write(line)
        except urllib.error.URLError as error:
            print(line, error)
        except urllib.error.HTTPError as error:
            print(line, error.code)

    v.close

def main():
    ...
```

Script: web requests

weather.py

```
import json
import urllib.request
from pprint import pprint

def get_local_weather():
    weather_base_url = 'http://forecast.weather.gov/MapClick.php?FcstType=json&'

    places = {
        'Austin': ['30.3074624', '-98.0335911'],
        'Portland': ['45.542094', '-122.9346037'],
        'NYC': ['40.7053111', '-74.258188']
    }

    for place in places:
        latitude = places[place][0]
        longitude = places[place][1]

        weather_url = weather_base_url + "lat=" + latitude + "&lon=" + longitude

        page_response = urllib.request.urlopen(weather_url).read()
        weather_data = json.loads(page_response.decode('utf-8'))

        forecast = parse_weather_data(weather_data)

        print("Today's date is", forecast['date'],
              "and the current temperature in", place, forecast['state'],
              "is", forecast['temp'], "degrees")

    def parse_weather_data(json_object):
        weather_obj = json_object

        todays_date = weather_obj['currentobservation']['Date']
        current_temp = weather_obj['currentobservation']['Temp']
        outlook = weather_obj['currentobservation']['Weather']
        state = weather_obj['currentobservation']['state']

        current_weather = {
            'date': todays_date,
            'temp': current_temp,
            'outlook': outlook,
            'state': state,
        }

        return current_weather

    def main():
        get_local_weather()
```

```
import json
import urllib.request
from pprint import pprint

def get_local_weather():
    weather_base_url = 'http://forecast.weather.gov/MapClick.php?FcstType=json&'

    places = {
        'Austin': ['30.3074624', '-98.0335911'],
        'Portland': ['45.542094', '-122.9346037'],
        'NYC': ['40.7053111', '-74.258188']
    }

    for place in places:
        latitude, longitude = places[place][0], places[place][1]
        weather_url = weather_base_url + "lat=" + latitude + "&lon=" + longitude

        page_response = urllib.request.urlopen(weather_url).read()
        weather_data = json.loads(page_response.decode('utf-8'))

        forecast = parse_weather_data(weather_data)

        print("Today's date is", forecast['date'],
              "and the current temperature in", place, forecast['state'],
              "is", forecast['temp'], "degrees")

def parse_weather_data(json_object):
    weather_obj = json_object

    todays_date = weather_obj['currentobservation']['Date']
    current_temp = weather_obj['currentobservation']['Temp']
    outlook = weather_obj['currentobservation']['Weather']
    state = weather_obj['currentobservation']['state']

    current_weather = {
        'date': todays_date,
        'temp': current_temp,
        'outlook': outlook,
        'state': state,
    }

    return current_weather
```

APIs

Some APIs you might look at later:

<http://sunlightfoundation.com/>

- lots of government data, state voting records, etc.

<https://developers.google.com/maps/documentation/static-maps/>

- create custom maps based on url parameters

<https://developers.google.com/apis-explorer/#p/>

- all of Google's other APIs

<https://www.flickr.com/services/api/>

<https://dev.twitter.com/>

What can you do with scripting?

- Math homework
- Ping urls
- Get the latest weather
-

What are some other ideas?

For later:

Hopefully, you said something like “Hello” and Brobot said something that sounded like a greeting in reply. For the “greet the robot” use case, we can use simple keyword matching, similar to how ELIZA and other early conversational UIs were modeled. Here’s the relevant code:

```
# Sentences we'll respond with if the user greeted us
GREETING_KEYWORDS = ("hello", "hi", "greetings", "sup", "what's up",)

GREETING_RESPONSES = [ "'sup bro", "hey", "*nods*", "hey you get my snap?" ]

def check_for_greeting(sentence):
    """If any of the words in the user's input was a greeting, return a greeting response"""
    for word in sentence.words:
        if word.lower() in GREETING_KEYWORDS:
            return random.choice(GREETING_RESPONSES)
```

This is the simplest possible implementation of a chatbot: it searches the user’s utterance for one or more known keywords and returns one of several possible responses. In practice you won’t want your bot to pick a truly random response—it’s better to cycle through a set of responses and avoid repeats. To keep the tutorial simple I’ve made Brobot completely stateless, so pure randomness will have to do.

Go ahead and modify the code above, right in the browser, to change Brobot’s behavior. Try returning only one response, or responding to more greetings. (If your code has an error, Brobot will pass along the Python message.)

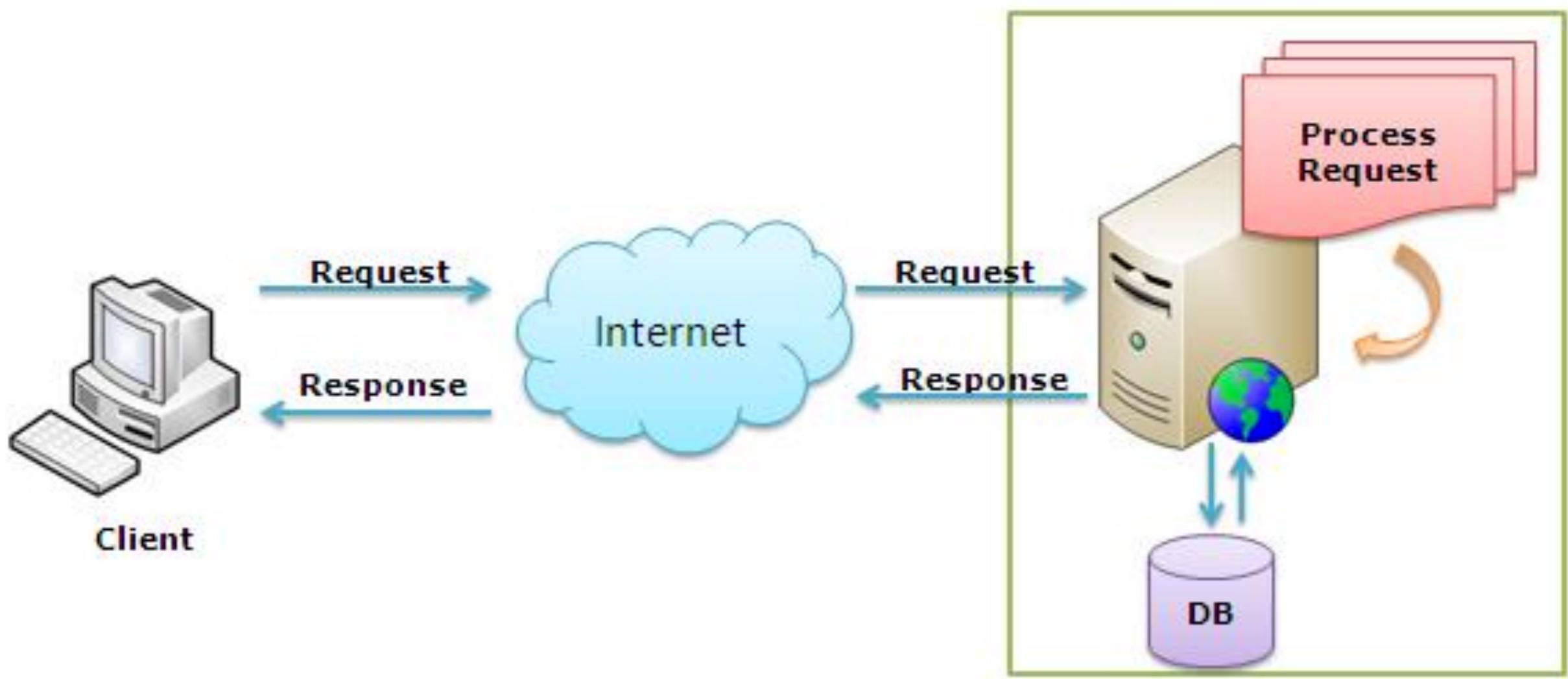
YOU:

Run code

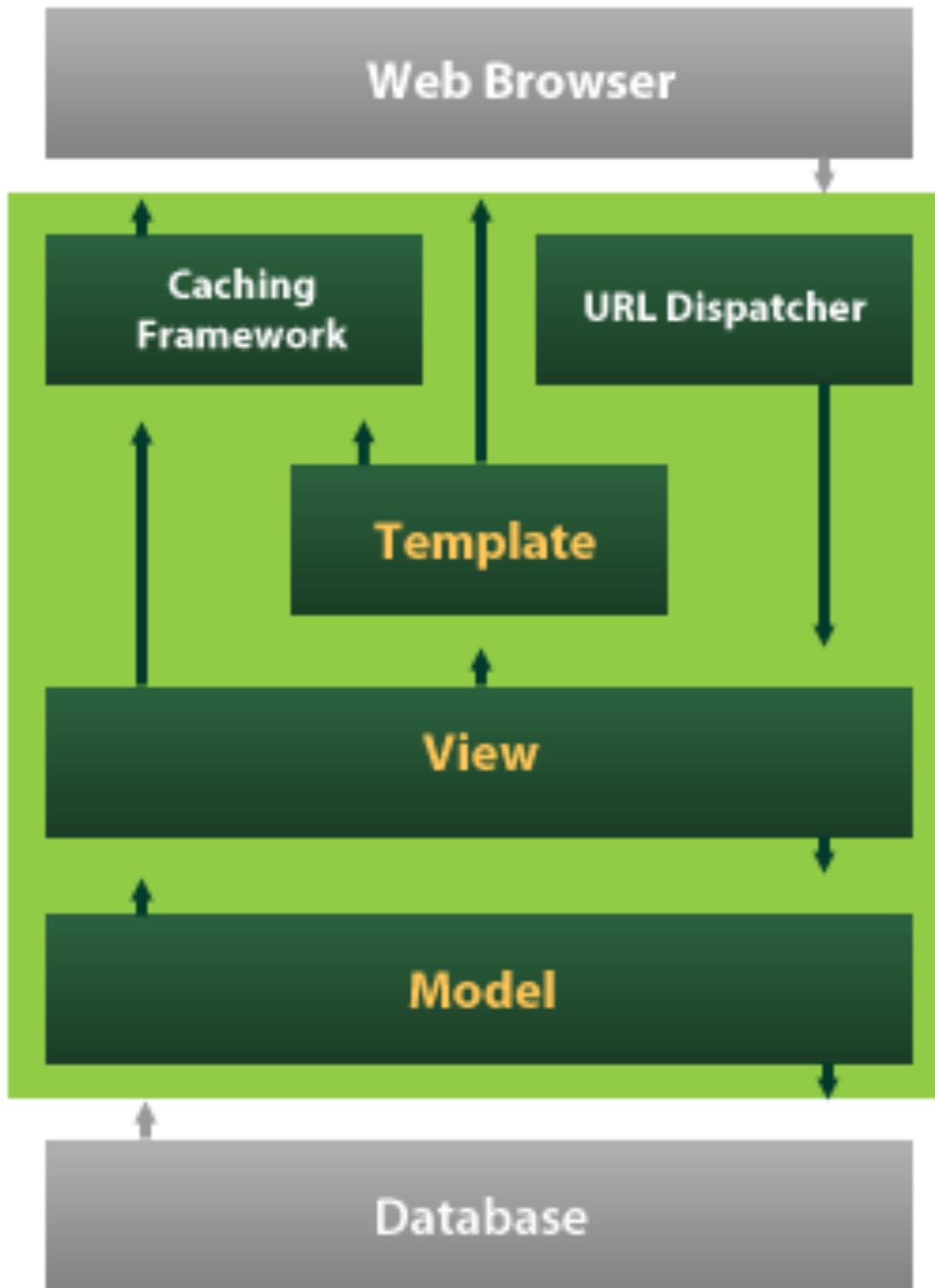
<https://apps.worldwritable.com/tutorials/chatbot/>

Let's build a web site

How the internet works



Frameworks



Frameworks

django

<https://www.djangoproject.com/>
<https://code.djangoproject.com/>

web2py

<http://www.web2py.com/>



Flask

web development,
one drop at a time

<http://flask.pocoo.org/>

Frameworks

- Source code (Python)
 - what the web site should do
- Templates (HTML)
 - how the web site should look

Resources:

<https://www.codecademy.com/en/tracks/htmlcss>

<http://www.w3schools.com/default.asp>



Host, run, and code Python in the cloud!

Get started for free. Our basic plan gives you access to machines with a full Python environment already installed. You can develop and host your website or any other code directly from your browser without having to install software or manage your own server.

Need more power? Upgraded plans start at \$4/month.

[Start running Python online in less than a minute! »](#)

[Watch our one-minute video »](#)

Not convinced? [Read what our users are saying!](#)

The screenshot shows the Pythonanywhere web interface. At the top, there's a navigation bar with the Pythonanywhere logo, a menu icon, and links for Send feedback, Forums, Help, Blog, Pricing & signup, and Log in. Below the header, the main content area has a large green button with the text "Start running Python online in less than a minute! »". Underneath it is a teal button with the text "Watch our one-minute video »". Further down, a message says "Not convinced? Read what our users are saying!". The main content area has a light blue background. It displays a success message: "All done! Your web app is now set up. Details below." Below this, there's a list of websites with checkmarks: "✓ www.tdd-django-tutorial.com", "✓ harry.pythonanywhere.com", and "✓ www.tester.com". To the right of this list is a button labeled "Reload www.bla.com". At the bottom, a note says "You can see your web app at <http://www.bla.com/>".



Plans and pricing

Beginner: Free!

A limited account with one web app at `your-username.pythonanywhere.com`, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython notebook support.

It works and it's a great way to get started!

[Create a Beginner account](#)

Education (coming soon)

Are you a teacher looking for a place your students can code Python? You're not alone. Click through to find out more about our [Education beta](#).

All of our paid plans come with a no-quibble 30-day money-back guarantee — you're billed monthly and you can cancel at any time. The minimum contract length is just one month. You get unrestricted Internet access from your applications, unlimited in-browser Python, Bash and database consoles, and full SSH access to your account. All accounts (including free ones) have screen-sharing with other PythonAnywhere accounts, and free SSL support (though you'll need to get a certificate for your own domains).

Hacker \$5/month

Run your Python code in the cloud from one web app and the console

A Python IDE in your browser with unlimited Python/bash consoles

One web app on your own domain or `your-`

Web dev \$12/month

If you want to host small Python-based websites for you or for your clients

A Python IDE in your browser with unlimited Python/bash consoles

Up to **2 web apps** on custom domains or

Startup \$99/month

Start a business and don't worry about having to scale to handle traffic spikes

A Python IDE in your browser with unlimited Python/bash consoles

Up to **3 web apps** on custom domains or

Custom \$5 to \$500/month

Want a combination that's not on the list? Create your own! All custom plans have:

A Python IDE in your browser with unlimited Python/bash consoles

Up to **20 web apps**, on custom domains or



Create your account

Username:

Email:

Password:

Password (again):

I agree to the [Terms and Conditions](#)

[Register](#)

We promise not to spam or pass your details on to anyone else.

Copyright © 2016 PythonAnywhere LLP — [Terms](#) — [Privacy](#)

"Python" is a registered trademark of the Python Software Foundation.

pythonanywhere.com

Send feedback Forums Help Blog Dashboard Account Log out
Unconfirmed email address

Consoles Files Web Schedule Databases

Thank you!

You're now signed up and logged in to your PythonAnywhere account: **barbaras**.

We've sent an email to **bshaurette@outlook.com**. If you click the link in the email to confirm your email address, we'll be able to reset your password if you forget it in the future.

What next?

We've got some helpers to get you started with some common tasks — why not try one out? Alternatively, if you don't want any help, just click the "X" button above and to the right.

- I want to start learning Python
- I want to follow the Django Tutorial
- I want to create a web application
- I have built a web app on my local PC and want to deploy it on PythonAnywhere
- I want to clone and hack on my GitHub project
- I want to check out the PythonAnywhere Education features

If there's something else you think we should have here, [click here to let us know](#). You can access these task helpers again at any time from the [Help](#) page

Start a new console:

Python: [3.4](#) / [3.3](#) / [2.7](#) / [2.6](#) IPython: [3.4](#) / [3.3](#) / [2.7](#) / [2.6](#) PyPy: [2.7](#)

0% used (0.00s of your 100 second CPU allowance)

Allowance resets in 23 hours, 59 minutes

Other: [Bash](#) | [MySQL](#)

Custom:

Your consoles:

You have no consoles. Click a link above to start one.



pythonanywhere.com



[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Dashboard](#) [Account](#) [Log out](#)

[Consoles](#) [Files](#) [Web](#) [Schedule](#) [Databases](#)

 [Add a new web app](#)

You have no web apps

To create a PythonAnywhere-hosted web app, click the "Add a new web app" button to the left.

Copyright © 2016 [PythonAnywhere LLP](#) — [Terms](#) — [Privacy](#)

"Python" is a registered trademark of the Python Software Foundation.

Create new web app



Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at flasksimpletutorial.pythonanywhere.com.

Want to change that? [Upgrade now!](#)

Otherwise, just click "Next" to continue.

Cancel

« Back

Next »



Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » [Django](#)
- » [web2py](#)
- » [Flask](#)
- » [Bottle](#)
- » [Manual configuration \(including virtualenvs\)](#)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

[Cancel](#)

[« Back](#)

[Next »](#)



Select a Python version

- » **Python 3.4 (Flask 0.10.1)**
- » **Python 3.3 (Flask 0.10.1)**
- » **Python 2.7 (Flask 0.9)**

Note: If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are [instructions here](#).

[Cancel](#)[« Back](#)[Next »](#)

Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app.
If this file already exists, its contents will be overwritten with the
new app.

Path

[Cancel](#)[« Back](#)[Next »](#)

All done! Your web app is now set up. Details below.



✓ flasksimplesetutorial.pythonanywhere.com

+ Add a new web app

Configuration for flasksimplesetutorial.pythonanywhere.com

Reload:

 Reload flasksimplesetutorial.pythonanywhere.com

Traffic:

How busy is your site?

This month (previous month)	0 (0)
Today (yesterday)	0 (0)
Hour (previous hour)	0 (0)

Want some more data? [Paying accounts](#) get pretty charts ;-)

Code:

What your site is running.

Source code: </home/flasksimplesetutorial/mysite>
WSGI configuration file: /var/www/flasksimplesetutorial_pythonanywhere_com_wsgi.py
Python version: 3.4

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to [Reload your web app](#) to activate it; NB - will do nothing if the virtualenv does not exist.

Enter path to a virtualenv, if desired

Log files:

The first place to look if something goes wrong.



Hello from Flask!

The screenshot shows a web browser window with the URL pythonanywhere.com in the address bar. The page displays the configuration for a web application named 'barbaras'. At the top, there are navigation icons for back, forward, and search. Below the address bar, the title 'Web app setup : barbaras : PythonAnywhere' is shown, along with a timestamp 'today (yesterday)'. To the right, the domain 'barbaras.pythonanywhere.com' is listed. On the far right, there are download, upload, and refresh icons.

Hour (previous hour) 1 (0)

Want some more data? [Paying accounts](#) get pretty charts ;-)

Code:

What your site is running.

Source code: </home/barbaras/mysite>

WSGI configuration file: /var/www/barbaras_pythonanywhere_com_wsgi.py

Python version: 3.4

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

Enter path to a virtualenv, if desired

Log files:

The first place to look if something goes wrong.

Access log: <barbaras.pythonanywhere.com.access.log>

Error log: <barbaras.pythonanywhere.com.error.log>

Server log: <barbaras.pythonanywhere.com.server.log>

Static files:

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
<i>Enter URL</i>	<i>Enter path</i>	



[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Dashboard](#) [Account](#) [Log out](#)

[Consoles](#) [Files](#) [Web](#) [Schedule](#) [Databases](#)

/> home > barbaras > mysite

[Open Bash console here](#)

[New](#)

[New](#)

[pycache_](#) /

Upload a file: no file selected

[flask_app.py](#) 2016-03-26 15:38 186 bytes

0% full (64.0 KB of your 512.0 MB quota)

Copyright © 2016 PythonAnywhere LLP — [Terms](#) — [Privacy](#)
"Python" is a registered trademark of the Python Software Foundation.

flask_app.py : /home/barbaras/mysite/flask_app.py : Editor : barbaras : PythonAnywhere

pythonanywhere.com

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal Save Save as... >>> ⌘

```
1 |
2 # A very simple Flask Hello World app for you to get started with...
3
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10     return 'Hello from Flask!'
11
12
```

>>> Run this file

flask_app.py : /home/barbaras/mysite/flask_app.py : Editor : barbaras : PythonAnywhere

pythonanywhere.com

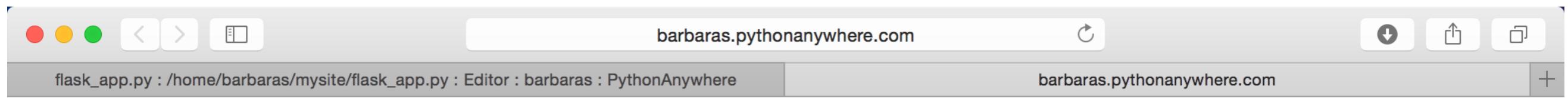
Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal Save Save as... >>> ⌘

```
1 # A very simple Flask Hello World app for you to get started with...
2
3 from flask import Flask
4
5 app = Flask(__name__)
6
7 @app.route('/')
8 def hello_world():
9     r = "Hello from Barbara!"
10
11 return r
```

>>> Run this file



Hello from Barbara!

A screenshot of the PythonAnywhere Files interface. The URL in the address bar is pythonanywhere.com. The top navigation bar includes links for Send feedback, Forums, Help, Blog, Dashboard, Account, and Log out. Below the navigation is a horizontal menu with tabs: Consoles, Files (which is selected and highlighted in blue), Web, Schedule, and Databases. The current path is displayed as / > home > barbaras > mysite. On the right, there is a link to Open Bash console here. The main area shows a file tree with a new directory entry 'Enter new directory name' and a 'New' button. Another 'Enter new file name' field and 'New' button are shown. The directory structure includes a folder named 'pycache_/' and another folder named 'templates/'. A file named 'flask_app.py' is listed with details: Upload date 2016-05-04 18:43, size 1.1 KB, and download, edit, and delete icons. A message at the bottom right indicates the quota status: 0% full (72.0 KB of your 512.0 MB quota).

A screenshot of the PythonAnywhere Files interface, similar to the one above but showing a different directory structure. The URL in the address bar is pythonanywhere.com. The top navigation bar and menu are identical. The current path is / > home > barbaras > mysite > templates. The main area shows a file tree with 'Enter new directory name' and 'New' buttons. An 'Enter new file name' field and 'New' button are also present. The directory structure includes a file named 'main_page.html'. A message at the bottom right indicates the quota status: 0% full (72.0 KB of your 512.0 MB quota). At the very bottom, there is a footer with copyright information: Copyright © 2016 PythonAnywhere LLP — Terms — Privacy and "Python" is a registered trademark of the Python Software Foundation.



/> home > barbaras > mysite > templates > main_page.html

Keyboard shortcuts: Normal

1

```
1 |
```

```
<html>
<head>
<title>My comments page</title>
</head>

<body>

<div>This is the first placeholder comment.</div>
<div>This is the second placeholder comment. It's no more interesting than the
first.</div>
<div>This is the third placeholder comment. It's actually quite exciting!</div>

<div>
<form action="." method="POST">
<textarea name="contents" placeholder="Enter a comment"></textarea>
<input type="submit" value="Post comment">
</form>
</div>

</body>
</html>
```

pythonanywhere.com

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > templates > main_page.html

Keyboard shortcuts: Normal Save Save as... ⌘

```
1 <html>
2 <head>
3 <title>My comments page</title>
4 </head>
5
6 <body>
7
8 <div>This is the first placeholder comment.</div>
9 <div>This is the second placeholder comment. It's no more interesting than the first.</div>
10 <div>This is the third placeholder comment. It's actually quite exciting!</div>
11
12 <div>
13 <form action"." method="POST">
14 <textarea name="contents" placeholder="Enter a comment"></textarea>
15 <input type="submit" value="Post comment">
16 </form>
17 </div>
18
19 </body>
20 </html>
```

main_page.html : /home/barbaras/mysite/templates/main... flask_app.py : /home/barbaras/mysite/flask_app.py : E... barbaras.pythonanywhere.com +

 pythonanywhere

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal ▾ Save Save as... >>> ⌘

```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4 app.config["DEBUG"] = True
5
6 @app.route('/')
7 def index():
8     return render_template("main_page.html")
9
```

>>> Run this file

[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Dashboard](#) [Account](#) [Log out](#)[Consoles](#) [Files](#)[Web](#)[Schedule](#)[Databases](#)[✓ barbaras.pythonanywhere.com](#)[+ Add a new web app](#)

Configuration for [barbaras.pythonanywhere.com](#)

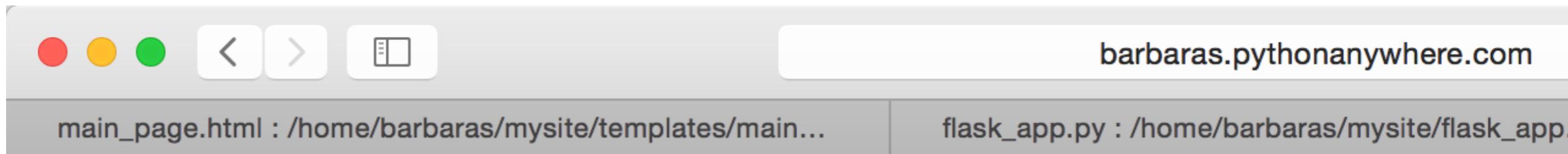
Reload:

 Reload
barbaras.pythonanywhere.com

A green rectangular button with white text. It contains a circular arrow icon followed by the word "Reload" and the URL "barbaras.pythonanywhere.com".

Best before date:

Free sites have a limited lifespan, but you can renew that here up to a maximum of three months from today's date. You can always extend it later too! We'll send you an email a week before it expires. [See here for more details.](#)



This is the first placeholder comment.

This is the second placeholder comment. It's no more interesting than the first.

This is the third placeholder comment. It's actually quite exciting!

Enter a comment

Post comment

main_page.html : /home/barbaras/mysite/templates/main... flask_app.py : /home/barbaras/mysite/flask_app.py : E... My comments page +

 pythonanywhere

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > templates > main_page.html

Keyboard shortcuts: Normal ▲ Save Save as... ▾

```
1 <html>
2 <head>
3 <title>My comments page</title>
4 </head>
5
6 <body>
7
8 <div>This is the first placeholder comment.</div>
9 <div>This is the second placeholder comment. It's no more interesting than the first.</div>
10 <div>This is the third placeholder comment. It's actually quite exciting!</div>
11
12 <div>
13 <form action="." method="POST">
14 <textarea name="contents" placeholder="Enter a comment"></textarea>
15 <input type="submit" value="Post comment">
16 </form>
17 </div>
18
19 </body>
20 </html>
```

```
<html>
<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
        integrity="sha512-dTfge/zgoMYpP7QbHy4gWMEGsbsdZeCXz7irItjcC3sPUFtf0kuFbDz/ixG7ArTxmDjLXDmezHubeNikyKGVyQ=="
        crossorigin="anonymous">

    <title>My comments page</title>
</head>

<body>

    <nav class="navbar navbar-inverse">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false"
                    aria-controls="navbar">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="#">My scratchpad</a>
            </div>
        </div>
    </nav>

    <div class="container">

        <div>This is the first placeholder comment.</div>
        <div>This is the second placeholder comment. It's no more interesting than the first.</div>
        <div>This is the third placeholder comment. It's actually quite exciting!</div>

        <div class="row">
            <form action"." method="POST">
                <textarea class="form-control" name="contents" placeholder="Enter a comment"></textarea>
                <input type="submit" value="Post comment">
            </form>
        </div>

    </div>

</body>
</html>
```



/ > home > barbaras > mysite > templates > main_page.html

```
1 <html>
2 <head>
3 |
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
8         integrity="sha512-dTfge/zgoMYpP7QbHy4gWMEGbsdZeCXz7irItjcC3sPUFtf0kuFbDz/ixG7ArTxmDjLXDmezHubeNikyKGVYQ==" crossorigin="anonymous">
9
10 <title>My comments page</title>
11 </head>
12
13 <body>
14
15     <nav class="navbar navbar-inverse">
16         <div class="container">
17             <div class="navbar-header">
18                 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false"
19                     aria-controls="navbar">
20                     <span class="sr-only">Toggle navigation</span>
21                     <span class="icon-bar"></span>
22                     <span class="icon-bar"></span>
23                     <span class="icon-bar"></span>
24                 </button>
25                 <a class="navbar-brand" href="#">My scratchpad</a>
26             </div>
27         </div>
28     </nav>
29
30     <div class="container">
31
32         <div class="row">This is the first placeholder comment.</div>
33         <div class="row">This is the second placeholder comment. It's no more interesting than the first.</div>
34         <div class="row">This is the third placeholder comment. It's actually quite exciting!</div>
35
36         <div class="row">
37             <form action"." method="POST">
38                 <textarea class="form-control" name="contents" placeholder="Enter a comment"></textarea>
39                 <input type="submit" value="Post comment">
40             </form>
41         </div>
42
43     </div>
44
45 </body>
46 </html>
```

This screenshot shows a web browser window with the URL `barbaras.pythonanywhere.com` in the address bar. The main content area is titled "My scratchpad". Inside, there are three placeholder comments:

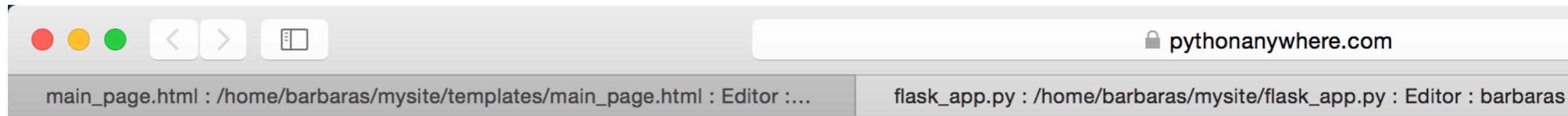
- This is the first placeholder comment.
- This is the second placeholder comment. It's no more interesting than the first.
- This is the third placeholder comment. It's actually quite exciting!

Below these comments is a text input field with the placeholder text "Enter a comment". At the bottom left is a button labeled "Post comment".

Sending and receiving data

Method Not Allowed

The method is not allowed for the requested URL.



```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4 app.config["DEBUG"] = True
5
6 @app.route('/', methods=['GET', "POST"])
7 def index():
8     return render_template("main_page.html")
9
```



main_page.html : /home/barbaras/mysite/templates/main_page.html : Editor : ...

flask_app.py : /home/barbaras/mysite/flask_app.py : Editor : barbaras : Py... X

My scratchpad

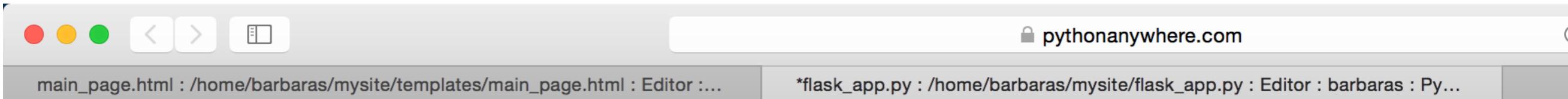
This is the first placeholder comment.

This is the second placeholder comment. It's no more interesting than the first.

This is the third placeholder comment. It's actually quite exciting!

Enter a comment

Post comment



The screenshot shows a web browser window with the PythonAnywhere logo at the top left. The address bar indicates the URL is pythonanywhere.com. There are two tabs open: "main_page.html : /home/barbaras/mysite/templates/main_page.html : Editor ..." and "*flask_app.py : /home/barbaras/mysite/flask_app.py : Editor : barbaras : Py...".



/> home > barbaras > mysite > flask_app.py *

Se

Key

```
1 from flask import Flask, redirect, render_template, request, url_for
2
3 app = Flask(__name__)
4 app.config["DEBUG"] = True
5
6 comments = []
7
8 @app.route('/', methods=["GET", "POST"])
9 def index():
10     if request.method == "GET":
11         return render_template("main_page.html", comments=comments)
12
13     comments.append(request.form["contents"])
14     return redirect(url_for('index'))
```

```
from flask import Flask, redirect, render_template, request, url_for

app = Flask(__name__)
app.config["DEBUG"] = True

comments = []

@app.route('/', methods=["GET", "POST"])
def index():
    if request.method == "GET":
        return render_template("main_page.html", comments=comments)

    comments.append(request.form["contents"])
    return redirect(url_for('index'))
```



main_page.html : /home/barbaras/mysite/templates/main_page.html : Editor : ...

flask_app.py : /home/barbaras/mysite/flask_app.py : Editor : barbaras : Py... X

My scratchpad

This is the first placeholder comment.

This is the second placeholder comment. It's no more interesting than the first.

This is the third placeholder comment. It's actually quite exciting!

Enter a comment

Post comment



/ > home > barbaras > mysite > templates > main_page.html

```
9  
10 <title>My comments page</title>  
11 </head>  
12  
13 <body>  
14  
15 <nav class="navbar navbar-inverse">  
16 <div class="container">  
17 <div class="navbar-header">  
18 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" data-bbox="118 468 198 508" aria-controls="navbar">  
19 <span class="sr-only">Toggle navigation</span>  
20 <span class="icon-bar"></span>  
21 <span class="icon-bar"></span>  
22 <span class="icon-bar"></span>  
23 </button>  
24 <a class="navbar-brand" href="#">My scratchpad</a>  
25 </div>  
26 </div>  
27 </div>  
28 </nav>  
29  
30 <div class="container">  
31  
32 {% for comment in comments %}  
33 <div class="row">  
34 {{ comment }}  
35 </div>  
36 {% endfor %}  
37  
38 <div class="row">  
39 <form action="" method="POST">
```

```
<html>
<head>

<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
integrity="sha512-dTfge/zgoMYpP7QbHy4gWMEGsbsdZeCXz7irItjcC3sPUFtf0kuFbDz/ixG7ArTxmDjLXDmezHubeNikyKGVyQ==" crossorigin="anonymous">

<title>My comments page</title>
</head>

<body>

<nav class="navbar navbar-inverse">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false"
aria-controls="navbar">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
<a class="navbar-brand" href="#">My scratchpad</a>
</div>
</div>
</nav>

<div class="container">

{%
  for comment in comments %}
  <div class="row">
    {{ comment }}
  </div>
{% endfor %}

<div class="row">
  <form action"." method="POST">
    <textarea class="form-control" name="contents" placeholder="Enter a comment"></textarea>
    <input type="submit" value="Post comment">
  </form>
</div>

</div>

</body>
</html>
```



main_page.html : /home/barbaras/mysite/templates/main_page.html : Editor :...

flask_a

My scratchpad

This is a whole new comment!

This is another new comment!

Enter a comment

Post comment

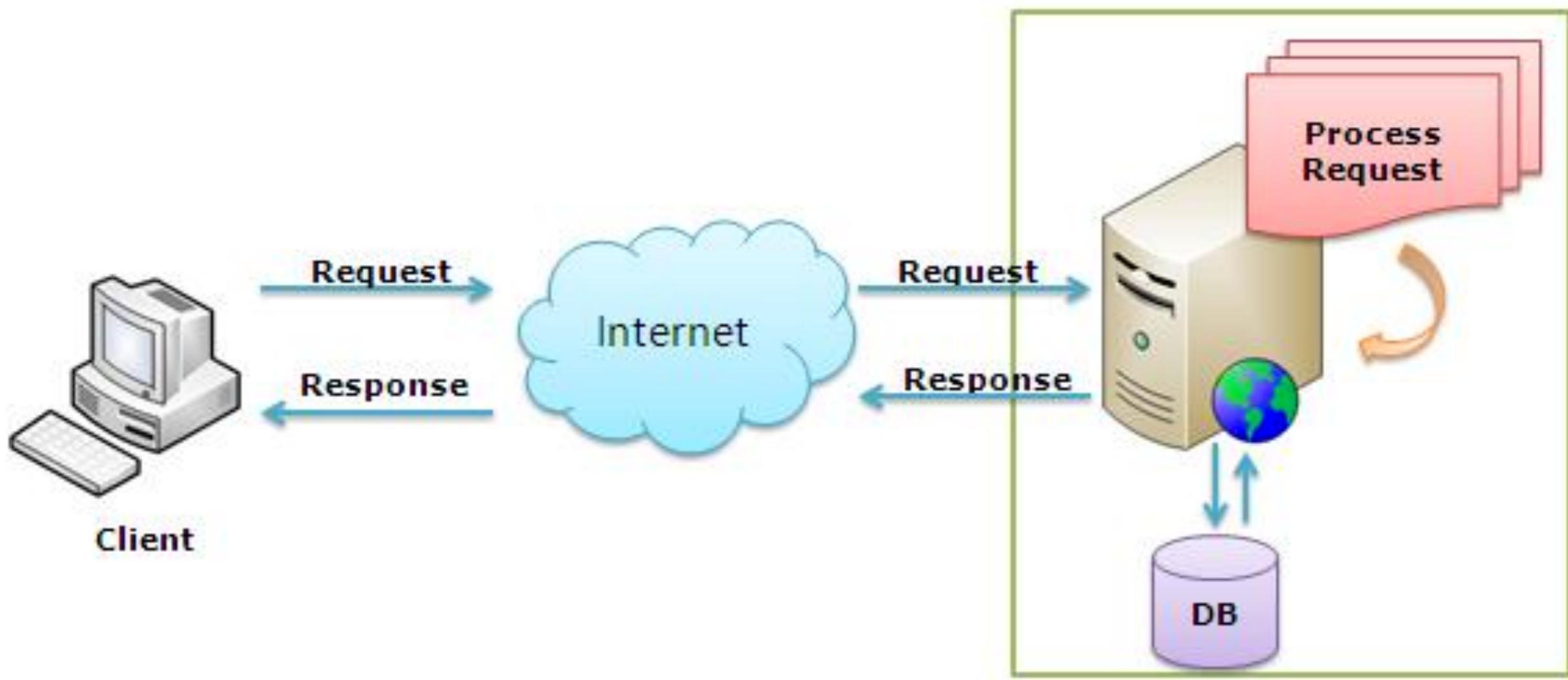
Bring on the database!

What is a database?

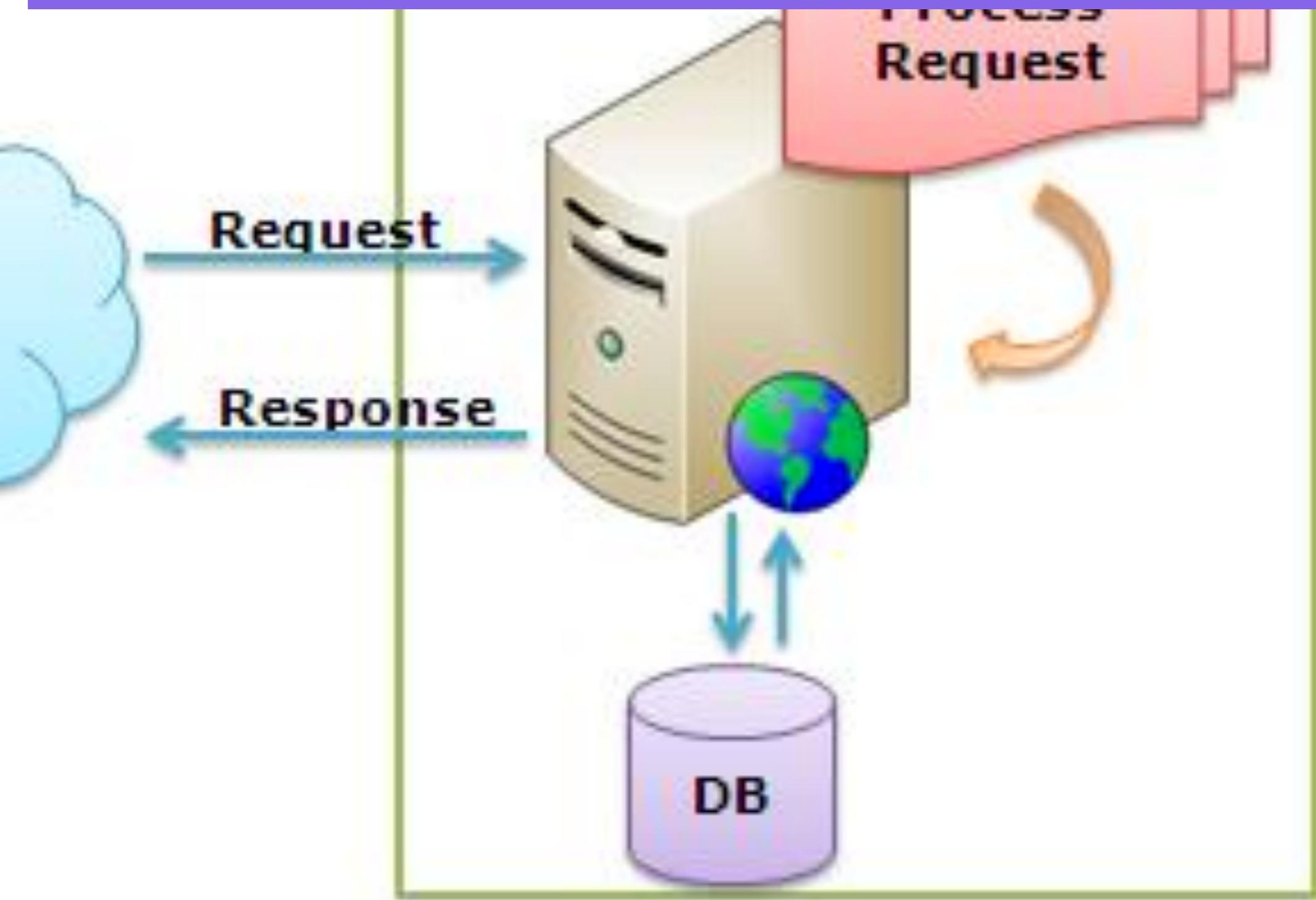
A program that stores data.

- Files
- Tables
- Text records
- Images
- Other kinds of objects

Bring on the database!



Bring on the database!





[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Dashboard](#) [Account](#) [Log out](#)

[Consoles](#) [Files](#) [Web](#) [Schedule](#)

[Databases](#)

[MySQL](#)

[Postgres](#)

MySQL settings

Connecting:

Use these settings in your web applications.

Database host address: `barbaras.mysql.pythonanywhere-services.com`

Username: `barbaras`

Your databases:

You don't have any databases yet. To start using MySQL, enter a new password in the form below, and note it down: you'll need it to access the databases once you've created them.

MySQL password:

This should be different to your main PythonAnywhere password, because it is likely to appear in plain text in any web applications you write.

New password:

Confirm password:

[Set MySQL password](#)

Your MySQL password has been changed.

[MySQL](#)[Postgres](#)

MySQL settings

Connecting:

Use these settings in your web applications.

Database host address: `flasksimpletutorial.mysql.pythonanywhere-services.com`

Username: `flasksimpletutor`

Your databases:

Click a database's name to start a MySQL console logged in to it.

Start a console on: `flasksimpletutor$default`

Create a database

Your database names always start with your username + "\$". There's no need to type that prefix in below, though: PythonAnywhere will automatically add it.

Database name:

MySQL password:

This should be different to your main PythonAnywhere password, because it is likely to appear in plain text in any web applications you write.

New password:

Confirm password:

Create a database

Your database names always start with your username + "\$". There's no need to type that prefix in below, though: PythonAnywhere will automatically add it.

Database name:

[Create](#)

Your databases:

Click a database's name to start a MySQL console logged in to it.

Start a console on: [flasksimpletutor\\$comments](#)

Start a console on: [flasksimpletutor\\$default](#)

main_page.html : /home/barbaras/mysit... flask_app.py : /home/barbaras/mysite/fl... Database Settings: PythonAnywhere My comments page +

 pythonanywhere

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal ▾ Save Save as... >>> ⌘

```
1 from flask import Flask, redirect, render_template, request, url_for
2 from flask.ext.sqlalchemy import SQLAlchemy
3
4 app = Flask(__name__)
5 app.config["DEBUG"] = True
6
7 SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}{}".format(
8     username=<the username from the 'Databases' tab>,
9     password=<the password you set on the 'Databases' tab>,
10    hostname=<the database host address from the 'Databases' tab>,
11    database_name=<the database name you chose, probably yourusername$comments>,
12 )
13 app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
14 app.config["SQLALCHEMY_POOL_RECYCLE"] = 299
15
16 db = SQLAlchemy(app)
17
18 comments = []
19
20 @app.route('/', methods=["GET", "POST"])
21 def index():
22     if request.method == "GET":
23         return render_template("main_page.html", comments=comments)
24
25     comments.append(request.form["contents"])
26     return redirect(url_for('index'))
```

>>> Run this file

[Consoles](#)[Files](#)[Web](#)[Schedule](#)[Databases](#)[MySQL](#)[Postgres](#)

MySQL settings

Connecting:

Use these settings in your web applications.

Database host address: `barbaras.mysql.pythonanywhere-services.com`

Username: `barbaras`

Your databases:

Click a database's name to start a MySQL console logged in to it.

Start a console on: [barbaras\\$default](#)

Start a console on: [barbaras\\$myflasksample](#)

```
from flask import Flask, redirect, render_template, request, url_for
from flask.ext.sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config["DEBUG"] = True

SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}/{"
    .format(
        username="barbaras",
        password="????",
        hostname="barbaras.mysql.pythonanywhere-services.com",
        databasename="barbaras$myflasksample",
    )
app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
app.config["SQLALCHEMY_POOL_RECYCLE"] = 299

db = SQLAlchemy(app)

comments = []

@app.route('/', methods=["GET", "POST"])
def index():
    if request.method == "GET":
        return render_template("main_page.html", comments=comments)

    comments.append(request.form["contents"])
    return redirect(url_for('index'))
```

All done! Your web app is now set up. Details below.



✓ flasksimpletutorial.pythonanywhere.com

+ [Add a new web app](#)

Configuration for [flasksimpletutorial.pythonanywhere.com](#)

Reload:

 [Reload flasksimpletutorial.pythonanywhere.com](#)

Traffic:

How busy is your site?

This month (previous month)	0 (0)
Today (yesterday)	0 (0)
Hour (previous hour)	0 (0)



main_page.html : /home/barbaras/mysite/templates/main_page.html : Editor : ...

flask_app.py :

My scratchpad

Enter a comment

Post comment

pythonanywhere.com

main_page.html : /home/barbaras/mysit... flask_app.py : /home/barbaras/mysite/fl... Database Settings: PythonAnywhere More... My comments page +

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal Save Save as... >>>

```
1 from flask import Flask, redirect, render_template, request, url_for
2 from flask.ext.sqlalchemy import SQLAlchemy
3
4 app = Flask(__name__)
5 app.config["DEBUG"] = True
6
7 SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}{}".format(
8     username="<the username from the 'Databases' tab>",
9     password="<the password you set on the 'Databases' tab>",
10    hostname="<the database host address from the 'Databases' tab>",
11    database="<the database name you chose, probably yourusername$comments>",
12 )
13 app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
14 app.config["SQLALCHEMY_POOL_RECYCLE"] = 299
15
16 db = SQLAlchemy(app)
17
18 class Comment(db.Model):
19
20     __tablename__ = "comments"
21
22     id = db.Column(db.Integer, primary_key=True)
23     content = db.Column(db.String(4096))
24
25 comments = []
26
```

>>> Run this file

Untitled spreadsheet



File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive



fx

id

A

B

C

D

E

1

content

1 This is the first placeholder comment.

2 This is the second placeholder comment. It's no more interesting than the first.

3 This is the third placeholder comment. It's actually quite exciting!

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19



comments ▾

```
from flask import Flask, redirect, render_template, request, url_for
from flask.ext.sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config["DEBUG"] = True

SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}/{"
    .format(
        username="barbaras",
        password="???" ,
        hostname="barbaras.mysql.pythonanywhere-services.com",
        database="barbaras$myflasksample",
    )
app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
app.config["SQLALCHEMY_POOL_RECYCLE"] = 299

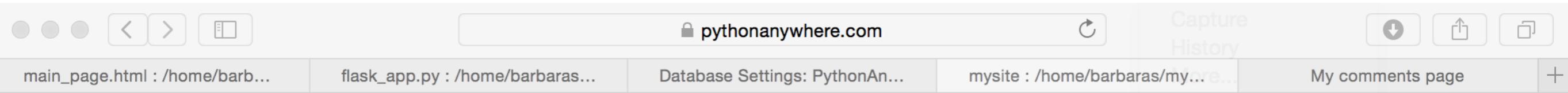
db = SQLAlchemy(app)

class Comment(db.Model):
    __tablename__ = "comments"
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(4096))

comments = []

@app.route('/', methods=["GET", "POST"])
def index():
    if request.method == "GET":
        return render_template("main_page.html", comments=comments)

    comments.append(request.form["contents"])
    return redirect(url_for('index'))
```

[Consoles](#)[Files](#)[Web](#)[Schedule](#)[Databases](#)[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Dashboard](#) [Account](#) [Log out](#)[Check for updates](#)
[Quit](#)

/ > home > barbaras > mysite

[Open Bash console here](#)[New](#)[New](#) [__pycache__/](#) [templates/](#)

Upload a file: [Choose File](#) no file selected

[flask_app.py](#) 2016-03-28 18:36 1.0 KB

0% full (176.0 KB of your 512.0 MB quota)

Copyright © 2016 PythonAnywhere LLP — [Terms](#) — [Privacy](#)

"Python" is a registered trademark of the Python Software Foundation.

main_page.html : /home/barb... flask_app.py : /home/barbaras... Database Settings: PythonAn... Bash console 2588467 : bar... My comments page +

 pythonanywhere

Bash console 2588467 [Share with others](#)

```
18:47 ~/mysite $ ipython3.4
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
Type "copyright", "credits" or "license" for more information.

IPython 4.1.2 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: from flask_app import db
In [2]: db.create_all()
In [3]: 
```

Your databases:

Click a database's name to start a MySQL console logged in to it.

Start a console on: [barbaras\\$default](#)

Start a console on: [barbaras\\$myflasksample](#)

main_page.html : /home/barbaras... flask_app.py : /home/barbaras/m... MySQL: barbaras\$myflasksampl... Bash console 2588467 : barbar... My comments page +

 pythonanywhere

MySQL: barbaras\$myflasksample

Send feedback Forums Help Blog Dashboard Account Log out

Share with others

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 19428
Server version: 5.5.42-log Source distribution

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
+-----+
| Tables_in_barbaras$myflasksample |
+-----+
| comments
+-----+
1 row in set (0.00 sec)

mysql> describe comments;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| id    | int(11)   | NO  | PRI | NULL    | auto_increment |
| content | varchar(4096) | YES |     | NULL    |             |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> 
```

Bring on the database!

So now we have:

- code in our Flask app to connect to the database
- a Python definition of what we want to store in the database
- tables in the database to store the data

Let's add some code to use all of that!

main_page.html : /home/barbaras... flask_app.py : /home/barbaras/m... MySQL: barbaras\$myflasksampl... Bash console 2588467 : barbaras... My comments page +

 pythonanywhere

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal Save Save as... >>> 

```
6
7     SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}{}".format(
8         username="barbaras",
9         password="blutrose9",
10        hostname="barbaras.mysql.pythonanywhere-services.com",
11        database="barbaras$myflasksample",
12    )
13    app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
14    app.config["SQLALCHEMY_POOL_RECYCLE"] = 299
15
16    db = SQLAlchemy(app)
17
18    class Comment(db.Model):
19        __tablename__ = "comments"
20        id = db.Column(db.Integer, primary_key=True)
21        content = db.Column(db.String(4096))
22
23    # comments = [] ## delete this, or just comment it out
24
25    @app.route('/', methods=["GET", "POST"])
26    def index():
27        if request.method == "GET":
28            # return render_template("main_page.html", comments=comments) ## remove this line, replace with the one below
29            return render_template("main_page.html", comments=Comment.query.all())
30
31    comments.append(request.form["contents"])
32    return redirect(url_for('index'))
33
```

>>> Run this file

pythonanywhere.com

main_page.html : /home/barbaras... flask_app.py : /home/barbaras/m... MySQL: barbaras\$myflasksampl... Bash console 2588467 : barbar... My comments page +

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > templates > main_page.html

Keyboard shortcuts: Normal Save Save as... ⌘

```
8     integrity="sha512-dTfge/zgoMYpP7QbHy4gWMEGsbsdZeCXz7irItjcC3sPUFtf0kuFbDz/ixG7ArTxmDjLXDmezHubeNikyKGVyQ==" crossorigin="anonymous">
9
10 <title>My comments page</title>
11 </head>
12
13 <body>
14
15   <nav class="navbar navbar-inverse">
16     <div class="container">
17       <div class="navbar-header">
18         <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false"
19             aria-controls="navbar">
20           <span class="sr-only">Toggle navigation</span>
21           <span class="icon-bar"></span>
22           <span class="icon-bar"></span>
23           <span class="icon-bar"></span>
24         </button>
25         <a class="navbar-brand" href="#">My scratchpad</a>
26       </div>
27     </div>
28   </nav>
29
30   <div class="container">
31
32     {% for comment in comments %}
33       <div class="row">
34         {{ comment.content }}
35       </div>
36     {% endfor %}
37
38   <div class="row">
39     <form action"." method="POST">
40       <textarea class="form-control" name="contents" placeholder="Enter a comment"></textarea>
41       <input type="submit" value="Post comment">
42     </form>
43   </div>
44
45 </div>
46
47 </body>
```

main_page.html : /home/barbaras... flask_app.py : /home/barbaras/m... MySQL: barbaras\$myflasksampl... Bash console 2588467 : barbaras... My comments page +

 pythonanywhere

Send feedback Forums Help Blog Dashboard Account Log out

/> home > barbaras > mysite > flask_app.py

Keyboard shortcuts: Normal Save Save as... >>> 

```
7 SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}/{name}".format(
8     username="barbaras",
9     password="blutrose9",
10    hostname="barbaras.mysql.pythonanywhere-services.com",
11    database="barbaras$myflasksample",
12 )
13 app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
14 app.config["SQLALCHEMY_POOL_RECYCLE"] = 299
15
16 db = SQLAlchemy(app)
17
18 class Comment(db.Model):
19     __tablename__ = "comments"
20     id = db.Column(db.Integer, primary_key=True)
21     content = db.Column(db.String(4096))
22
23 @app.route('/', methods=["GET", "POST"])
24 def index():
25     if request.method == "GET":
26         return render_template("main_page.html", comments=Comment.query.all())
27
28     # comments.append(request.form["contents"]) ## remove this line, replace with code below
29     comment = Comment(content=request.form["contents"])
30     db.session.add(comment)
31     db.session.commit()
32
33     return redirect(url_for('index'))
34
```

>>> Run this file

```
from flask import Flask, redirect, render_template, request, url_for
from flask.ext.sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config["DEBUG"] = True

SQLALCHEMY_DATABASE_URI = "mysql+mysqlconnector://{}:{}@{}{}".format(
    username="barbaras",
    password="???",
    hostname="barbaras.mysql.pythonanywhere-services.com",
    database="barbaras$myflasksample",
)
app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
app.config["SQLALCHEMY_POOL_RECYCLE"] = 299

db = SQLAlchemy(app)

class Comment(db.Model):
    __tablename__ = "comments"
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(4096))

@app.route('/', methods=["GET", "POST"])
def index():
    if request.method == "GET":
        return render_template("main_page.html", comments=Comment.query.all())

    comment = Comment(content=request.form["contents"])
    db.session.add(comment)
    db.session.commit()

    return redirect(url_for('index'))
```



main_page.html : /home/barbaras...

flask_app.py : /home/barbaras/m...

My scratchpad

This is my latest comment.

Enter a comment

Post comment

main_page.html : /home/barbaras... flask_app.py : /home/barbaras/m... MySQL: barbaras\$myflasksampl... Bash console 2588467 : barbaras... My comments page +

 pythonanywhere

Send feedback Forums Help Blog Dashboard Account Log out

MySQL: barbaras\$myflasksample

Share with others

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> show tables;
+-----+
| Tables_in_barbaras$myflasksample |
+-----+
| comments
+-----+
1 row in set (0.00 sec)

mysql> describe comments;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| id    | int(11)   | NO  | PRI | NULL    | auto_increment |
| content | varchar(4096) | YES |     | NULL    |             |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from comments;
+-----+
| id | content          |
+-----+
| 1  | This is my latest comment. |
+-----+
1 row in set (0.01 sec)

mysql>
```

What can YOU do with a web site?

- Make a game
- Build a blog
-
-

What are some other ideas?

For later:



Versions

[Development \(unstable\)](#)

[Flask 0.10.x \(stable\)](#)

[Flask 0.9.x](#)

Related Topics

[Documentation overview](#)

- [Previous: Quickstart](#)
- [Next: Introducing Flaskr](#)

Quick search

Go

Tutorial

You want to develop an application with Python and Flask? Here you have the chance to learn that by example. In this tutorial we will create a simple microblog application. It only supports one user that can create text-only entries and there are no feeds or comments, but it still features everything you need to get started. We will use Flask and SQLite as database which comes out of the box with Python, so there is nothing else you need.

If you want the full sourcecode in advance or for comparison, check out the [example source](#).

- [Introducing Flaskr](#)
- [Step 0: Creating The Folders](#)
- [Step 1: Database Schema](#)
- [Step 2: Application Setup Code](#)
- [Step 3: Creating The Database](#)
- [Step 4: Request Database Connections](#)
- [Step 5: The View Functions](#)
 - [Show Entries](#)
 - [Add New Entry](#)
 - [Login and Logout](#)
- [Step 6: The Templates](#)
 - [layout.html](#)
 - [show_entries.html](#)
 - [login.html](#)
- [Step 7: Adding Style](#)
- [Bonus: Testing the Application](#)

Let's make a game!

Open a new window (File > New File) and type these lines:

```
secret_number = 7  
  
guess = input("What number am I thinking of? ")  
  
if secret_number == guess:  
    print("Yay! You got it.")  
else:  
    print("No, that's not it.")
```

Choose Run > Run Module. Save to Desktop as 'guess.py'.

Games!

Open a new window (File > New File) and type these lines:

```
from random import randint  
  
secret_number = randint(1, 10)  
  
while True:  
    guess = input("What number am I thinking of? ")  
  
    if secret_number == guess:  
        print("Yay! You got it.")  
        break  
    else:  
        print("No, that's not it.")
```

Choose Run > Run Module. Save to Desktop as 'guess2.py'.

Games!

Open a new window (File > New File) and type these lines:

```
from random import randint

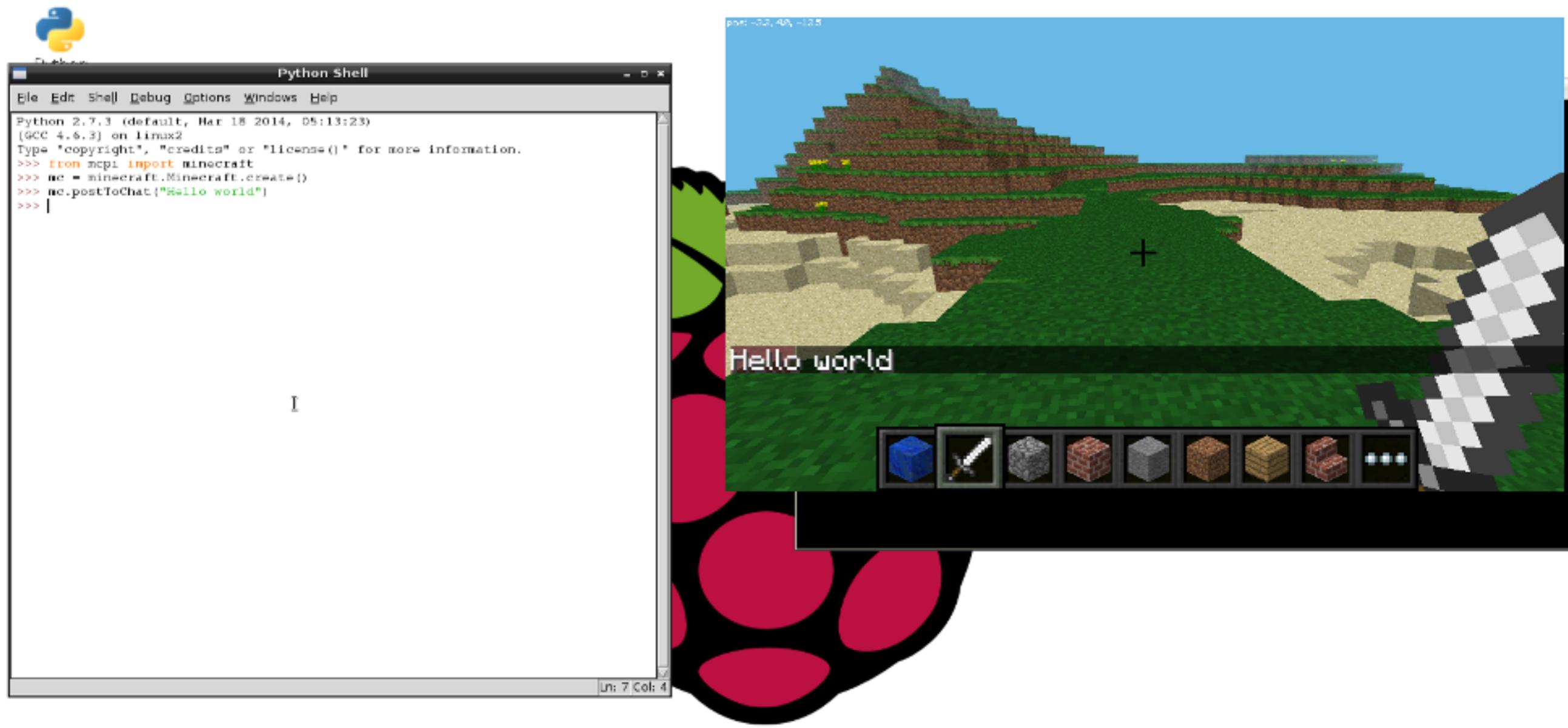
secret_number = randint(1, 10)

while True:
    guess = input("What number am I thinking of? ")

    if secret_number == guess:
        print("Yay! You got it.")
        break
    elif secret_number > guess:
        print("No, that's too low.")
    else:
        print("No, that's too high.")
```

Choose Run > Run Module. Save to Desktop as 'guess3.py'.

Minecraft!



Minecraft!

```
>>> from mcpi import minecraft  
>>> mc = minecraft.Minecraft.create()  
  
>>> mc.postToChat("Hello world")  
  
>>> pos = mc.player.getPos()  
>>> print pos.x, pos.y, pos.z  
  
>>> mc.player.setPos(pos.x, pos.y+100, pos.z)  
  
>>> mc.setBlock(pos.x+1, pos.y, pos.z, 1)
```

(Air: 0, Grass: 2, Dirt: 3)

Minecraft!

```
>>> from mcpi import block  
  
>>> dirt = block.DIRT.id  
>>> mc.setBlock(x, y, z, dirt)  
  
>>> stone = block.STONE.id  
>>> mc.setBlocks(x+1, y+1, z+1, x+11, y+11, z+11, stone)  
  
>>> tnt = 46  
>>> mc.setBlocks(x+1, y+1, z+1, x+11, y+11, z+11, tnt)  
>>> mc.setBlocks(x+1, y+1, z+1, x+11, y+11, z+11, tnt, 1)
```

Minecraft!

Want to do even weirder stuff?

First, make sure you've done this at the top of a file:

```
import time  
import random  
from mcpi.minecraft import Minecraft, blocks  
  
mc = Minecraft.create()
```

Minecraft!

Want to do even weirder stuff?

```
def nyan_cat():
    """Generates a constant trail of wool blocks of various
    colors underneath the player."""
    while True:
        x, y, z = mc.player.getPos()
        # Pick a random color for the wool.
        color = random.randint(1, 15)
        # Create a wool block of that color underneath where the player is
        # standing.
        mc.setBlock(x, y - 1, z, blocks.WOOL, color)
        time.sleep(0.1)
```

Minecraft!

Want to do even weirder stuff?

```
def turn_blocks_to_diamonds():
    """Listens for any block events (right-clicks on blocks while
holding the sword) and converts any such block to a diamond block."""
    while True:
        # Each time this is called, it will return a list of all the events
        # that happened since the last time you called it.
        events = mc.events.pollBlockHits()
        for event in events:
            # Each event points to the position of the block that was clicked
            # on.
            x, y, z = event.pos
            mc.setBlock(x, y, z, blocks.DIAMOND_BLOCK)
        time.sleep(0.1)
```

Minecraft!

Want to do even weirder stuff?

```
def get_random_safe_place():
    """Returns the X, Y, and Z coordinates for a random
    "safe" place in the game world."""
    # Worlds in MCPI are small. The X and Z coordinates range from -128 to
    # 128.
    x, z = [random.randint(-128, 128) for i in range(2)]
    # Use that X/Z coordinate pair to find a Y-height that is guaranteed to
    # be safe.
    y = mc.getHeight(x, z)
    return [x, y, z]
```

Then we can use this function to do this...

Minecraft!

```
def tree_teleportation():
    """
    All trees in your world now have magic powers.

    (Teleport to a random place whenever you right-click
    with a sword on a tree block.)
    """

    while True:
        events = mc.events.pollBlockHits()
        for event in events:
            x, y, z = event.pos
            # Find the block at that location.
            block = mc.getBlockWithData(x, y, z)
            # All varieties of wood have the same block ID.
            if block.id == blocks.WOOD:
                x, y, z = get_random_place()
                mc.player.setPos(x, y, z)
                time.sleep(0.1)
```

Congratulations!
You're now a Pythonista!

What else can you do with Python?

- Make more games
- Edit music and videos
- Build web sites
- Write a program that does your homework for you ...
- Teach a robot how to move
- What are some other ideas?

Reference

Find these slides at:

<https://github.com/mechanicalgirl/young-coders-tutorial>

(choose 2016, then Intermediate)

Raspberry Pi

Help setting up your new computer:

<http://www.raspberrypi.org/quick-start-guide>

Minecraft on your new Raspberry Pi:

<https://www.raspberrypi.org/learning/getting-started-with-minecraft-pi/worksheet/>

<http://www.stuffaboutcode.com/p/minecraft-api-reference.html>