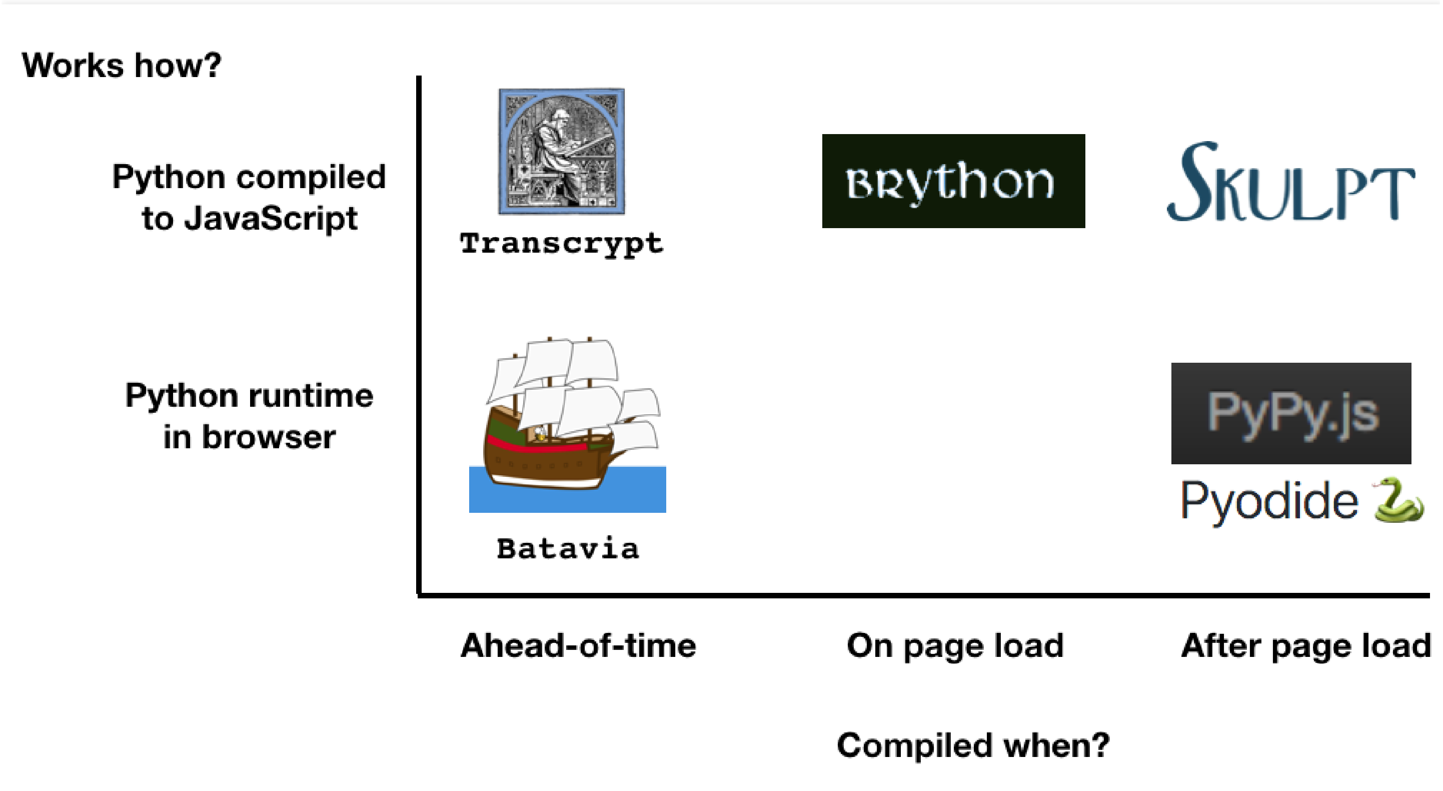
Run Python in the Web Browser

1. downloading and installing python
2. many different environments

---

1. an environment in the browser, just go ahead and write some code.
2. write web applications 100% in Python



Y-axis : **How is the Python being run by the browser?**
translate the Python code to Javascript and then run that
implement a Python interpreter in the browser and pass your Python through it

X-axis **When does the Python get compiled?**
compile ahead-of-time
on the page load
after the page has loaded

1. Transcrypt: compiling ahead-of-time

Transcrypt is a compiler that translates Python into JavaScript ahead-of-time.

https://github.com/QQuick/Transcrypt

https://transcrypt.org/docs/html/index.html

Python 3.7 to JavaScript compiler

- Precompiled into highly readable, efficient JavaScript, downloads kB's rather than MB's
- Generates JavaScript for humans
- Also runs on top of node.js

pip install transcrypt

transcrypt -n hello.py

http-server

2. Brython: Python script tags

Brython is a compiler that's written in JavaScript

https://github.com/brython-dev/brython

Brython (Browser Python) is an implementation of Python 3 running in the browser

To use Brython, all there is to do is:

1. Load the script brython.js.
2. Run the function brython() on page load, like <body onload="brython()">.
3. Write Python code inside tags <script type="text/python">.

```
<head>
    <script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/brython/3.7.5/brython.min.js">
    </script>
</head>

<body onload=brython()>
    <script type="text/python">
    from browser import document, alert
    def greet(event):
        alert("Hello " + document["name-box"].value + "!")
    document["greet-button"].bind("click", greet)
    </script>

    <input id="name-box" placeholder="Enter your name">
    <button id="greet-button">Say Hello</button>
</body>
```

3. Skulpt: interactive Python in the browser

https://github.com/skulpt/skulpt

Skulpt is a Javascript implementation of the Python programming language

Skulpt was originally written for education, to provide an interactive Python environment in the browser.

```html
<script src="https://cdn.jsdelivr.net/npm/skulpt@1.2.0/dist/skulpt.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/skulpt@1.2.0/dist/skulpt-stdlib.js" type="text/javascript"></script>
```

```javascript
<script type="text/javascript">
// output functions are configurable.  This one just appends some text
// to a pre element.
function outf(text) {
    var mypre = document.getElementById("output");
    mypre.innerHTML = mypre.innerHTML + text;
}
function builtinRead(x) {
    if (Sk.builtinFiles === undefined || Sk.builtinFiles["files"][x] === undefined)
            throw "File not found: '" + x + "'";
    return Sk.builtinFiles["files"][x];
}

// Here's everything you need to run a python program in skulpt
// grab the code from your textarea
// get a reference to your pre element for output
// configure the output function
// call Sk.importMainWithBody()
function runit() {
   var prog = document.getElementById("yourcode").value;
   var mypre = document.getElementById("output");
   mypre.innerHTML = '';
   Sk.pre = "output";
   Sk.configure({output:outf, read:builtinRead});
   var myPromise = Sk.misceval.asyncToPromise(function() {
       return Sk.importMainWithBody("<stdin>", false, prog, true);
   });
}
```

```
</script>
```

4. PyPy.js: a full Python interpreter in your web browser

https://github.com/pypyjs/pypyjs

PyPy. Compiled into JavaScript. JIT-compiling to JavaScript at runtime.

recompiled PyPy to JavaScript using emscripten

built a just-in-time Python-to-JavaScript compiler for compiling commonly-used code

```
<script src="./pypyjs.js"></script>
```

```
<script type="text/javascript">
  pypyjs.exec(
    // Run some Python
    'y = [x**2. for x in range(10)]'
  ).then(function() {
    // Transfer the value of y from Python to JavaScript
    return pypyjs.get('y');
  }).then(function(result) {
    // Display an alert box with the value of y in it
    alert(result);
  });
</script>
```

5. Batavia: A lightweight bytecode interpreter

https://github.com/beeware/batavia

A JavaScript implementation of the Python virtual machine.

https://batavia.readthedocs.io/en/latest/tutorial/index.html

The idea behind Batavia is to take Python bytecode, and ship it to a browser where it runs in a JavaScript implementation of the Python virtual machine.

```
cd testserver
pip install -r requirements.txt
./manage.py runserver
http://localhost:8000
```

```
<script id="batavia-customcode" type="application/python-bytecode">
```

```
7gwNCkIUE1cWAAAA4wAAAAAAAAAAAAAAAIAAABAAAAAcw4AAABlAABkAACDAQABZAEAUykCegtI
ZWxsbyBXb3JsZE4pAdoFcHJpbnSpAHICAAAAcgIAAAD6PC92YXIvZm9sZGVycy85cC9uenY0MGxf
OTc0ZGRocDFoZnJjY2JwdzgwMDAwZ24vVC90bXB4amMzZXJyddoIPG1vZHVsZT4BAAAAcwAAAAA=
    </script>
```

6. Pyodide: Like a Jupyter Notebook in your web browser

https://github.com/pyodide/pyodide

Pyodide is a Python distribution for the browser and Node.js based on WebAssembly
Pyodide is a port of CPython to WebAssembly/Emscripten.
many general-purpose packages : scientific Python packages including NumPy, pandas, SciPy, Matplotlib, and scikit-learn.
a full environment, a bit like a Jupyter Notebook, but it runs entirely in your browser. data science pipelines fully in the client
Pyodide comes with a robust Javascript ⟺ Python foreign function interface so that you can freely mix these two languages in your code with minimal friction.

https://alpha.iodide.io/notebooks/300/

```
<html>
  <head>
      <script src="
https://cdn.jsdelivr.net/pyodide/v0.20.0/full/pyodide.js
"></script>
  </head>
  <body>
    Pyodide test page <br>
    Open your browser console to see Pyodide output
    <script type="text/javascript">
      async function main(){
        let pyodide = await loadPyodide();
        console.log(pyodide.runPython(`
            import sys
            sys.version
        `));
        console.log(pyodide.runPython("print(1 + 2)"));
      }
```

```
        main();
      </script>
    </body>
  </html>
```

System	When compiled	How Python is run	Extra features	Typical use case	Built-in DOM manipulation	Size of download
Transcrypt	Ahead-of-time	Transpiled to JS		Replacement for JS	Yes	42 kB
Brython	On page load	Transpiled to JS		Replacement for JS	Yes	152 kB
Skulpt	Just-in-time	Transpiled to JS		Python environment in browser	No	130 kB (intepreter) + 150 kB (stdlib)
PyPy.js	Just-in-time	PyPy interpreter in JavaScript	Multiple interpreters, virtual filesystem, calling JavaScript from Python	Python environment in browser	Yes	12 MB
Batavia	Ahead-of-time	Custom bytecode interpreter in JavaScript		Python environment in browser	Yes	400 kB
Pyodide	Just-in-time	CPython interpreter in WebAssembly	NumPy, SciPy, Pandas, Matplotlib	Data science pipelines in browser	Yes	~10MB, more if you're using the libraries

https://anvil.works/blog/python-in-the-browser-talk

https://github.com/Michael8968/python-browser

https://stromberg.dnsalias.org/~strombrg/pybrowser/python-browser.html