The Ultimate List of Equivalents in Python and JavaScript by @DjangoTricks

Time-honored	Python	JavaScript (ECMAScript 5)
Parse integer	`number = int(text)`	`number = parseInt(text, 10);`
Conditional assignment	`value = 'ADULT' if age >= 18 else 'CHILD'`	`value = age >= 18? 'ADULT': 'CHILD';`

Object attribute value by name	`attribute = 'color'`
	`value = getattr(obj, attribute, 'GREEN')`
	`setattr(obj, attribute, value)`
		`attribute = 'color';`
		`value = obj[attribute] || 'GREEN';`
		`obj[attribute] = value;`

Dictionary value by key	`key = 'color'`
	`value = dictionary.get(key, 'GREEN')`
	`dictionary[key] = value`
		`key = 'color';`
		`value = dictionary[key] || 'GREEN';`
		`dictionary[key] = value;`

List slices	`items = [1, 2, 3, 4, 5]`
	`first_two = items[:2]  # [1, 2]`
	`last_two = items[-2:]  # [4, 5]`
	`middle_three = items[1:4]  # [2, 3, 4]`
		`items = [1, 2, 3, 4, 5];`
		`first_two = items.slice(0, 2);  // [1, 2]`
		`last_two = items.slice(-2);  // [4, 5]`
		`middle_three = items.slice(1, 4);  // [2, 3, 4]`

String slices	`text = 'ABCDE'`
	`first_two = text[:2]  # 'AB'`
	`last_two = text[-2:]  # 'DE'`
	`middle_three = text[1:4]  # 'BCD'`
		`text = 'ABCDE';`
		`first_two = text.slice(0, 2);  // 'AB'`
		`last_two = text.slice(-2);  // 'DE'`
		`middle_three = text.slice(1, 4); // 'BCD'`

List operations	`items1 = ['A']`
	`items2 = ['B']`
	`items = items1 + items2  # items == ['A', 'B']`
	`items.append('C')  # ['A', 'B', 'C']`
	`items.insert(0, 'D')  # ['D', 'A', 'B', 'C']`
	`first = items.pop(0)  # ['A', 'B', 'C']`
	`last = items.pop()  # ['A', 'B']`
	`items.delete(0)  # ['B']`
		`items1 = ['A'];`
		`items2 = ['B'];`
		`items = items1.concat(items2);  // items === ['A', 'B']`
		`items.push('C');  // ['A', 'B', 'C']`
		`items.unshift('D');  // ['D', 'A', 'B', 'C']`
		`first = items.shift();  // ['A', 'B', 'C']`
		`last = items.pop();  // ['A', 'B']`
		`items.splice(0, 1);  // ['B']`

Joining lists of strings	`items = ['A', 'B', 'C']` `text = ', '.join(items)  # 'A, B, C'`	`items = ['A', 'B', 'C'];` `text = items.join(', ');  // 'A, B, C'`

JSON	`import json`
	`json_data = json.dumps(dictionary, indent=4)`
	`dictionary = json.loads(json_data)`
		`json_data = JSON.stringify(dictionary, null, 4);`
		`dictionary = JSON.parse(json_data);`

Splitting strings by regular expressions	`import re`
	`# One or more of "!?." followed by whitespace`
	`delimiter = re.compile(r'[!?\.]+\s*')`
	`text = "Hello!!! What's new? Follow me."`
	`sentences = delimiter.split(text)`
	`# sentences == ['Hello', "What's new", 'Follow me', '']`
		`// One or more of "!?." followed by whitespace`
		`delimiter = /[!?\.]+\s*/;`
		`text = "Hello!!! What's new? Follow me.";`
		`sentences = text.split(delimiter);`
		`// sentences == ["Hello", "What's new", "Follow me", ""]`

Matching regular expression patterns in strings	`import re`
	`# name, "@", and domain`
	`pattern = re.compile(`
	`  r'([\w.+\-]+)@([\w\-]+\.[\w\-.]+)'`
	`)`
	`match = pattern.match('hi@example.com')`
	`# match.group(0) == 'hi@example.com'`
	`# match.group(1) == 'hi'`
	`# match.group(2) == 'example.com'`
	
	`text = 'Say hi at hi@example.com'`
	`first_match = pattern.search(text)`
	`if first_match:`
	`  start = first_match.start()  # start == 10`
		`// name, "@", and domain`
		`pattern = /([\w.+\-]+)@([\w\-]+\.[\w\-.]+)/;`
		
		`match = 'hi@example.com'.match(pattern);`
		`// match[0] === 'hi@example.com'`
		`// match[1] === 'hi'`
		`// match[2] === 'example.com'`
		
		`text = 'Say hi at hi@example.com';`
		`first_match = text.search(pattern);`
		`if (first_match > -1) {`
		`  start = first_match;  // start === 10`
		`}`

Replacing patterns in strings using regular expressions	`import re`
	`# name, "@", and domain`
	`pattern = re.compile(`
	`  r'([\w.+\-]+)@([\w\-]+\.[\w\-.]+)'`
	`)`
	
	`html = pattern.sub(`
	`  r'<a href="mailto:\g<0>">\g<0></a>',`
	`  'Say hi at hi@example.com',`
	`)`
	`# html == 'Say hi at <a href="mailto:hi@example.com">hi@example.com</a>'`
	
	`text = pattern.sub(`
	`  lambda match: match.group(0).upper(),`
	`  'Say hi at hi@example.com',`
	`)`
	`# text == 'Say hi at HI@EXAMPLE.COM'`
		`// name, "@", and domain`
		`pattern = /([\w.+\-]+)@([\w\-]+\.[\w\-.]+)/;`
		
		`html = 'Say hi at hi@example.com'.replace(`
		`  pattern,`
		`  '<a href="mailto:$&">$&</a>',`
		`);`
		`// html === 'Say hi at <a href="mailto:hi@example.com">hi@example.com</a>'`
		
		`text = 'Say hi at hi@example.com'.replace(`
		`  pattern,`
		`  function(match, p1, p2) {`
		`  return match.toUpperCase();`
		`  }`
		`);`
		`// text === 'Say hi at HI@EXAMPLE.COM'`

Error handling	`class MyException(Exception):`
	`  def __init__(self, message):`
	`  self.message = message`
	
	`  def __str__(self):`
	`  return self.message`
	
	`def proceed():`
	`  raise MyException('Error happened!')`
	
	`try:`
	`  proceed()`
	`except MyException as err:`
	`  print('Sorry! {}'.format(err))`
	`finally:`
	`  print('Finishing')`
		`function MyException(message) {`
		`  this.message = message;`
		`  this.toString = function() {`
		`  return this.message;`
		`  }`
		`}`
		
		`function proceed() {`
		`  throw new MyException('Error happened!');`
		`}`
		
		`try {`
		`  proceed();`
		`} catch (err) {`
		`  if (err instanceof MyException) {`
		`  console.log('Sorry! ' + err);`
		`  }`
		`} finally {`
		`  console.log('Finishing');`
		`}`

The Ultimate List of Equivalents in Python and JavaScript by @DjangoTricks

Future-proof	Python	JavaScript (ECMAScript 6)
Variables in strings	`name = 'World'` `value = f"""Hello, {name}!` `Welcome!"""  # Python 3.6` `price = 14.9` `value = f'Price: {price:.2f} €'  # 'Price: 14.90 €'`	`name = 'World';` `value = `Hello, ${name}!` `Welcome!`;` `price = 14.9;` `value = `Price ${price.toFixed(2)} €`; // 'Price: 14.90 €'`

Unpacking lists	`a = 1`
	`b = 2`
	`a, b = b, a  # swap values`
	
	`first, second, *the_rest = [1, 2, 3, 4]  # Python 3.6`
	`# first == 1, second == 2, the_rest == [3, 4]`
		`a = 1;`
		`b = 2;`
		`[a, b] = [b, a];  // swap values`
		
		`[first, second, ...the_rest] = [1, 2, 3, 4];`
		`// first === 1, second === 2, the_rest === [3, 4]`

Lambda functions	`sum = lambda x, y: x + y` `square = lambda x: x ** 2`	`sum = (x, y) => x + y;` `square = x => Math.pow(x, 2);`

Iteration	`for item in ['A', 'B', 'C']:`
	`    print(item)`
		`for (let item of ['A', 'B', 'C']) {`
		`    console.log(item);`
		`}`

Generators	`def countdown(counter):`
	`    while counter > 0:`
	`        yield counter`
	`        counter -= 1`
	
	`for counter in countdown(10):`
	`    print(counter)`
		`function* countdown(counter) {`
		`    while (counter > 0) {`
		`        yield counter;`
		`        counter--;`
		`    }`
		`}`
		`for (let counter of countdown(10)) {`
		`    console.log(counter);`
		`}`

Sets	`s = set(['A'])`
	`s.add('B'); s.add('C')`
	`'A' in s`
	`len(s) == 3`
	`for elem in s:`
	`    print(elem)`
	
	`s.remove('C')`
		`s = new Set(['A']);`
		`s.add('B').add('C');`
		`s.has('A') === true;`
		`s.size === 3;`
		`for (let elem of s.values()) {`
		`    console.log(elem);`
		`}`
		`s.delete('C');`

Function arguments	`from pprint import pprint`
	
	`def create_post(**options):`
	`    pprint(options)`
	
	`def report(post_id, reason='not-relevant'):`
	`    pprint({'post_id': post_id, 'reason': reason})`
	
	`def add_tags(post_id, *tags):`
	`    pprint({'post_id': post_id, 'tags': tags})`
	
	`create_post(title='Hello, World!', content='')`
	`report(42)`
	`report(post_id=24, reason='spam')`
	`add_tags(42, 'python', 'javascript', 'django')`
		
		`function create_post(options) {`
		`    console.log(options);`
		`}`
		`function report(post_id, reason='not-relevant') {`
		`    console.log({post_id: post_id, reason: reason});`
		`}`
		
		`function add_tags(post_id, ...tags) {`
		`    console.log({post_id: post_id, tags: tags});`
		`}`
		
		`create_post({title: 'Hello, World!', content': ''});`
		`report(42);`
		`report(post_id=24, reason='spam');`
		`add_tags(42, 'python', 'javascript', 'django');`

Classes and inheritance	`class Post(object):`
	`    def __init__(self, id, title):`
	`        self.id = id`
	`        self.title = title`
	
	`    def __str__(self):`
	`        return self.title`
	
	`class Article(Post):`
	`    def __init__(self, id, title, content):`
	`        super(Article, self).__init__(id, title)`
	`        self.content = content`
	
	`class Link(Post):`
	`    def __init__(self, id, title, url):`
	`        super(Link, self).__init__(id, title)`
	`        self.url = url`
	
	`    def __str__(self):`
	`        return '{} ({})'.format(`
	`            super(Link, self).__str__(),`
	`            self.url,`
	`        )`
	
	`article = Article(1, 'Hello, World!',`
	`    'This is my first article.'`
	`)`
	`link = Link(2, 'The Example', 'http://example.com')`
	`# isinstance(article, Post) == True`
	`# isinstance(link, Post) == True`
	`print(link)`
	`# The Example (http://example.com)`
		`class Post {`
		`    constructor (id, title) {`
		`        this.id = id;`
		`        this.title = title;`
		`    }`
		`    toString() {`
		`        return this.title;`
		`    }`
		`}`
		
		`class Article extends Post {`
		`    constructor (id, title, content) {`
		`        super(id, title);`
		`        this.content = content;`
		`    }`
		`}`
		
		`class Link extends Post {`
		`    constructor (id, title, url) {`
		`        super(id, title);`
		`        this.url = url;`
		`    }`
		`    toString() {`
		`        return super.toString() + ' (' + this.url + ')';`
		`    }`
		`}`
		
		`article = new Article(1, 'Hello, World!',`
		`    'This is my first article.'`
		`);`
		`link = new Link(2, 'The Example', 'http://example.com');`
		`// article instanceof Post === true`
		`// link instanceof Post === true`
		`console.log(link.toString());`
		`// The Example (http://example.com)`

Class properties: getters and setters	`class Post(object):`
	`    def __init__(self, id, title):`
	`        self.id = id`
	`        self.title = title`
	`        self._slug = ''`
	
	`    @property`
	`    def slug(self):`
	`        return self._slug`
	
	`    @slug.setter`
	`    def slug(self, value):`
	`        self._slug = value`
	
	`post = Post(1, 'Hello, World!')`
	`post.slug = 'hello-world'`
	`print(post.slug)`
		`class Post {`
		`    constructor (id, title) {`
		`        this.id = id;`
		`        this.title = title;`
		`        this._slug = '';`
		`    }`
		
		`    set slug(value) {`
		`        this._slug = value;`
		`    }`
		
		`    get slug() {`
		`        return this._slug;`
		`    }`
		`}`
		
		`post = new Post(1, 'Hello, World!');`
		`post.slug = 'hello-world';`
		`console.log(post.slug);`

The Ultimate List of Equivalents in Python and JavaScript by @DjangoTricks

Bonus	Python	JavaScript (ECMAScript 6)

All truthful elements	`items = [1, 2, 3]`
	`all_truthy = all(items)`
	`# True`
		`items = [1, 2, 3];`
		`all_truthy = items.every(Boolean);`
		`// true`

Any truthful elements	`items = [0, 1, 2, 3]`
	`some_truthy = any(items)`
	`# True`
		`items = [0, 1, 2, 3];`
		`some_truthy = items.some(Boolean);`
		`// true`

Iterate through each element and its index	`items = ['a', 'b', 'c', 'd']`
	`for index, element in enumerate(items):`
	`  print(f'{index}: {element};')`
		`items = ['a', 'b', 'c', 'd'];`
		`items.forEach(function(element, index) {`
		`  console.log(`${index}: ${element};`);`
		`});`

Map elements to the results of a function	`items = [0, 1, 2, 3]`
	`all_doubled = list(`
	`  map(lambda x: 2 * x, items)`
	`)`
	`# [0, 2, 4, 6]`
		`items = [0, 1, 2, 3];`
		`all_doubled = items.map(`
		`  x => 2 * x`
		`);`
		`// [0, 2, 4, 6]`

Filter elements by a function	`items = [0, 1, 2, 3]`
	`only_even = list(`
	`  filter(lambda x: x % 2 == 0, items)`
	`)`
	`# [0, 2]`
		`items = [0, 1, 2, 3];`
		`only_even = items.filter(`
		`  x => x % 2 === 0`
		`);`
		`// [0, 2]`

Reduce elements by a function to a single value	`from functools import reduce`
	`items = [1, 2, 3, 4]`
	`total = reduce(`
	`  lambda total, current: total + current,`
	`  items,`
	`)`
	`# 10`
		`items = [1, 2, 3, 4];`
		`total = items.reduce(`
		`  (total, current) => total + current`
		`);`
		`// 10`

Merge dictionaries	`d1 = {'a': 'A', 'b': 'B'}`
	`d2 = {'a': 'AAA', 'c': 'CCC'}`
	`merged = {**d1, **d2}  # since Python 3.5`
	`# {'a': 'AAA', 'b': 'B', 'c': 'CCC'}`
		`d1 = {a: 'A', b: 'B'}`
		`d2 = {a: 'AAA', c: 'CCC'}`
		`merged = {...d1, ...d2};`
		`// {a: 'AAA', b: 'B', c: 'CCC'}`