

JAVASCRIPT

Python

```
// Variables
                                   # Variables
let fullName = "Jane Hacker";
                                   full_name = "Jane Hacker"
const pi = 3.14;
                                   pi = 3.14
// Objects (similar to "dicts")
                                   # dicts (similar to "Objects")
let translation = {
                                   translation = {
                                       "ola": "Hello",
    ola: "Hello",
    oi: "Hi",
                                       "oi": "hi",
};
// For loop
                                   # For loops
for (let name of names) \{
                                   for name in names:
    console.log("name:", name);
                                      print("name:", name)
                                   # While loops
// While loops
let x = 0;
                                   while x < 3:
while (x < 3) {
                                      print("X:", x)
    console.log("X:", x);
                                       x += 1
   x++:
// If-statements
                                   # If-statements
                                   if full_name == "Jane":
if (fullName === "Jane") {
                                      print("Hi, Jane!")
    console.log("Hi, Jane!");
                                   elif full_name == "Alice":
} else if (fullName === "Alice") {
                                      print("Hey Alice")
    console.log("Hey Alice");
} else {
                                      print("Don't know you")
    console.log("Don't know you");
                                   # List comprehension
// Array processing (map & filter)
                                   long_names = [
let longNames = names
                                       name.upper() for name in names
    .filter(n => n.length > 3)
                                       if len(name) > 3
    .map(n => n.toUpperCase());
// Functions
                                   # Functions
function greeter(name) {
                                   def greeter(name):
    console.log("Hi", name);
                                       print("Hi", name)
                                   greeter("Bob")
greeter("Bob");
                                   # Lambda function
// Arrow function expression
                                   dst = lambda x, y: x*x + y*y
const dst = (x, y) \Rightarrow x*x + y*y;
// Conjunctions
                                   # Conjunctions
if (age < 18 && drink === "beer") {if age < 18 and drink == "beer":
    console.log("Too young kiddo");
                                      print("Too young kiddo")
if (age > 18 || drink === "soda") {if age > 18 or drink == "soda":
                                      print("Great choice")
    console.log("Great choice");
                                   # Class syntax
// Class syntax
                                   class User(BaseUser):
class User extends BaseUser {
                                       def __init__(self, name):
    constructor(name) {
                                           self.name = name
        this.name = name;
                                           self.logged_in = False
        this.loggedIn = false;
                                      def log_in(self):
    logIn() {
                                           self.logged_in = True
        this.loggedIn = true;
let user = new User("jqhacker");
                                  user = User("jqhacker")
```

PYTHON FOR JS DEVELOPERS MORE PYTHON TYPES

```
sets "keys-only dict", with operations
a = {"a", 1, 4, "b"}
b = {"a", "b"}
print(a - b) # {1, 4}
```

tuples immutable list (array)

```
tup = ("a", "b", "c")
# can't do: tup["a"] = 3
```

Python features

```
template strings Python 3.6+ only
phrase = f"name is {full_name}"
```

Ternary operator

```
b = 3 \text{ if len(a)} > 3 \text{ else } 0
```

in operator can be used with any collection (dict, string, list, etc)

```
if "Paul" in names:
    print("Found Paul")
```

Try / except Code should throw exceptions when necessary, and use try/catch to handle errors

```
try:
    third_name = names[3]
except Exception as e:
    pass # Handle ANY exception
except IndexError as e:
    pass # Only one error type
```

GENERAL DIFFERENCES

blocking Callbacks aren't used in Python. The blocking approach is to pause execution and use return values. Although not commonly in use, Python 3.5+ supports async via await and async syntax.

object oriented programming

Python frameworks tend to use classic OOP more often than JS.

operator overloading Python allows custom classes to use operators.

Example: define a method called

__add__ to allow the + operator.

ZEN OF PYTHON

Simple is better than complex

There should only be one way to do it

Premature optimization is root of all evil