

Python Quick Ref

for JS Devs

ES6 & TypeScript to Python
Cheat Sheet & Quick Reference

JavaScript (ES6) / TypeScript

```
const users = ["Sam", "Alex"];
const user = { name: "Sam", age: 30 };

function greet(name) {
  return `Hello, ${name}!`;
}

users.forEach(u => console.log(greet(u)));

if (user.age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```



Python

```
users = ["Sam", "Alex"]
user = {"name": "Sam", "age": 30}

def greet(name):
    return f"Hello, {name}!"

for u in users:
    print(greet(u))

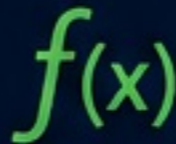
if user["age"] >= 18:
    print("Adult")
else:
    print("Minor")
```



Syntax
Comparison



Data Structures
& Collections



Functions
& Classes



Modules, APIs
& Async



Date, Time
& Regex



Best Practices
& Tips



Samir Solanki

[linkedin.com/in/samir38](https://www.linkedin.com/in/samir38)

Table of Contents



Syntax Overview



Variables



Data Types



Output & Comments



Code Blocks & Indentation



Operators



Imports & Modules



Overview Quick Ref



Collections Overview



Array vs List Basics



Array Methods

Table of Contents



	Array Iteration
	Object vs Dictionary
	Object Methods & Dict Methods
	Object Methods (Dict)
	Object Patterns & Best Practices
	Set in Python
	String Methods
	Functions
	Classes
	Typing
	Error Handling
	JSON
	API Calls & Networking
	Async Operations
	Date & Time
	Regular Expressions

ES6 / TypeScript vs Python

</> SYNTAX OVERVIEW

A high-level side-by-side look at the most fundamental syntax differences.

TS

ES6 / TypeScript

```
// Hello World in TypeScript
console.log("Hello, World!");
```

File Extension `.ts`

TypeScript files use `.ts`
(or `.tsx` for React)

Statement Ending

Statements end with a semicolon ;

```
const x = 10;
console.log(x);
```

Code Blocks

Blocks are defined with { }

```
if (x > 5) {
  console.log("x is greater");
} else {
  console.log("x is smaller");
}
```

Comments

```
// Single line comment
/*
  Multi-line comment
*/
```

Basic Structure

Organized with functions, classes,
and curly braces { }

```
function greet(name: string) {
  return `Hello, ${name}`;
}
```



Hello World



File Extension



Statement
Ending



Code
Blocks



Comments



Basic
Structure



Python

```
# Hello World in Python
print("Hello, World!")
```

File Extension `.py`

Python files use `.py`

Statement Ending

No semicolon ; needed

```
x = 10
print(x)
```

Code Blocks

Blocks are defined by indentation

```
if x > 5:
  print("x is greater")
else:
  print("x is smaller")
```

Comments

```
# Single line comment
"""
  Multi-line comment
"""
```

Basic Structure

Organized with indentation,
colons :, and blocks

```
def greet(name: str):
  return f"Hello, {name}"
```

KEY DIFFERENCES AT A GLANCE



Blocks

JS/TS use { }
Python uses
indentation



Semicolons

Required in JS/TS
Not needed
in Python



Syntax

JS/TS uses more
symbols
Python is cleaner
& readable



Comments

// or /* */
in JS/TS
or """ """
in Python



Files

`.ts` / `.tsx`
for TypeScript
`.py`
for Python



Philosophy

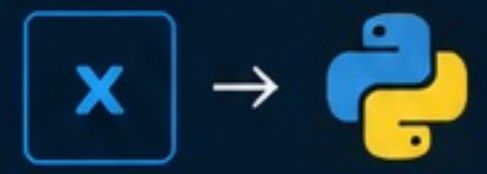
JS/TS: Flexible
Python: Readable
& Consistent



Remember: Python values readability and consistency.
Indentation is **not optional**—it's part of the syntax!

VARIABLES

ES6 / TypeScript vs Python



How to declare, assign and work with variables

TS ES6 / TypeScript

Basic Variable Assignment

```
1 let name = "John";
2 const age = 25;
3 var isActive = true;
4 let price = 99.99;
5 let items = [];
```

Declaration Keywords

let Block-scoped variable, can be reassigned

const Block-scoped constant, cannot be reassigned

var Function-scoped (avoid in modern JS/TS)

Type Annotations (Optional in TS)

```
let name: string = "John";
let age: number = 25;
let isActive: boolean = true;
let scores: number[] = [1, 2, 3];
let user: { id: number; name: string }
= { id: 1, name: "John" };
```

Reassignment

```
let count = 1;
count = 2; // allowed
const pi = 3.14;
// pi = 3.15; // X Error
```

Naming Conventions

- ✓ camelCase for variables and functions
- ✓ PascalCase for classes and types
- ✓ UPPER_CASE for constants
- ✓ \$ and _ are allowed
- ✓ Cannot start with a number

Examples:

userName, getData(), UserService, MAX_SIZE

VS



Both are dynamically typed – type is inferred at runtime.



Keep it simple!



Both are optional and used for better tooling & readability.



Python variables can always be reassigned.



Follow the idioms of the language.



Python

Basic Variable Assignment

```
1 name = "John"
2 age = 25
3 is_active = True
4 price = 99.99
5 items = []
```

No Keyword Needed

Python does not use let, const or var. Just use the variable name.

```
count = 0
message = "Hello"
is_valid = False
```

Type Hints (Optional in Python)

```
name: str = "John"
age: int = 25
is_active: bool = True
scores: list[int] = [1, 2, 3]
user: dict[str, int | str] = {
    "id": 1, "name": "John"
}
```

Reassignment

```
count = 1
count = 2 # allowed
pi = 3.14
pi = 3.15 # allowed
```

Naming Conventions (Pythonic)

- ✓ snake_case for variables and functions
- ✓ PascalCase for classes
- ✓ UPPER_CASE for constants
- ✓ _ is allowed, no \$
- ✓ Cannot start with a number

Examples:

user_name, get_data(), UserService, MAX_SIZE

QUICK COMPARISON

Feature	ES6 / TypeScript	Python
Declare variable	let / const / var	No keyword
Reassign	let (yes), const (no)	Always yes
Type system	Static (optional)	Dynamic (optional hints)
Block scope	let / const	Based on indentation
Constant	const	Convention: UPPER_CASE

COMMON GOTCHAS WHEN SWITCHING

- ✗ Don't use let / const / var in Python.
- ✗ No semicolons (;) at the end of statements.
- ✗ Indentation is required – it defines the block.
- ✗ Variables in Python are always mutable references.
- ✗ Use snake_case, not camelCase.



Thank You!

Thanks for reading!

I hope this [Python Quick Ref for JS Devs](#) helps you write better Python code and makes your transition smooth and enjoyable.

Keep learning, keep building,
and happy coding! 🚀



Keep Coding. Keep Growing.