**Python vs JavaScript** for core concepts: variables, if/else, loops, functions, and classes/objects—with **explanations + code + output**.

---

# 1. Variables: Python vs JavaScript

## Python

```
x = 10
name = "Alice"
```

- No keyword needed to declare variables
- Dynamically typed (type inferred automatically)
- No need for `let/const`

## JavaScript

```
let x = 10;
const name = "Alice";
```

- `let` → reassignable variable
- `const` → constant (cannot be reassigned)
- `var` (older, avoid in modern JS)

### Output (both languages conceptually)

```
x = 10
name = Alice
```

### Key Difference

| Python | JavaScript |
|---|---|
| No declaration keyword | Uses let, const |
| Simpler syntax | More control over scope |

---

# 2. If-Else Statement

## Python

```
x = 10

if x > 5:
    print("Greater")
else:
    print("Smaller")
```

### Output

```
Greater
```

---

## JavaScript

```
let x = 10;

if (x > 5) {
    console.log("Greater");
} else {
    console.log("Smaller");
}
```

### Output

```
Greater
```

### Key Differences

| Python | JavaScript |
| --- | --- |
| Uses indentation | Uses `{}` braces |
| No parentheses needed in condition | Requires parentheses |

---

# 3. Loops

## Python (for loop)

```
for i in range(3):
    print(i)
```

### Output

```
0
1
2
```

---

## JavaScript (for loop)

```
for (let i = 0; i < 3; i++) {
    console.log(i);
}
```

### Output

```
0
1
2
```

---

## Python while loop

```
i = 0
while i < 3:
    print(i)
    i += 1
```

## JavaScript while loop

```
let i = 0;
while (i < 3) {
    console.log(i);
    i++;
}
```

### Key Differences

| Python | JavaScript |
| --- | --- |
| `range()` used | Manual initialization |
| Cleaner syntax | More control, more verbose |

---

# 4. Functions

## Python

```
def add(a, b):
    return a + b

print(add(2, 3))
```

### Output

```
5
```

---

## JavaScript

```
function add(a, b) {
    return a + b;
}

console.log(add(2, 3));
```

### Output

```
5
```

### Modern JS (arrow function)

```
const add = (a, b) => a + b;
```

---

### Key Differences

| Python | JavaScript |
|---|---|
| `def` keyword | `function` keyword |
| Indentation-based block | `{}` block |
| No arrow function style originally | Arrow functions are common |

---

# 5. Classes & Objects

## Python Class

```
class Person:
    def __init__(self, name):
        self.name = name

    def greet(self):
        return "Hello " + self.name

p = Person("Alice")
print(p.greet())
```

### Output

```
Hello Alice
```

---

# JavaScript Class

```
class Person {
    constructor(name) {
        this.name = name;
    }

    greet() {
        return "Hello " + this.name;
    }
}

let p = new Person("Alice");
console.log(p.greet());
```

### Output

```
Hello Alice
```

---

### Key Differences

| Python | JavaScript |
| --- | --- |
| `__init__` constructor | `constructor()` |
| `self` required | `this` used |
| Simpler OOP syntax | Prototype-based internally |

---

# 6. Object Concept Difference (Important)

## Python

- Everything is an object (very consistent OOP model)
- Strong class-based structure

Example:

```
print(type(10))  # <class 'int'>
```

---

## JavaScript

- Prototype-based inheritance (though class syntax exists now)
- More flexible but historically more complex

Example:

```
console.log(typeof 10); // "number"
```

---

# FINAL SUMMARY

| Feature | Python | JavaScript |
|---|---|---|
| Variables | Simple, no keywords | let / const required |
| If-else | indentation-based | braces `{}` |
| Loops | range() simplifies loops | manual control |
| Functions | `def` keyword | function / arrow |
| Classes | clean OOP | class syntax over prototypes |

---

# QUICK REAL-WORLD DIFFERENCE

- **Python** → easier, cleaner, used in AI, data science, backend
- **JavaScript** → web development (frontend + backend with Node.js)