Python

JavaScript

Keywords

Python	JavaScript
`True, False, def, return, if, elif, else, and, or, not, assert, for, in, del, with, as`	`true, false, let, function, return, if, else, for, in, delete`

Types

Python	JavaScript

`bool`
`int`
`float`
`str`
`range`
	`Boolean`
	`Number`
	`String`

Literals

Python		JavaScript	
`bool`	`True, False`	`Boolean`	`true, false`
`int`	`1, 2, 17, 256`	`Number`	`13, 42, 3.14`
`float`	`3.14, 4.99`	`String`	`"Hello", 'Goodbye'`
`str`	`"Hello", 'Goodbye'`		

Operators

Python		JavaScript	
`Arithmetic`	`+, -, *, /, //, %`	`Arithmetic`	`+, -, *, **, /, %`
`String`	`+, *`	`String`	`+`
`Relational`	`<, <=, >, >=, ==, !=`	`Relational`	`<, <=, >, >=, ==, !=`
`Boolean`	`not, and, or`	`Boolean`	`!, &&, ||`

Variables

```python
# No explicit variable declaration
# Variables must be assigned before use
x = 12
y = 10 * x
```

```javascript
// Must declare variables before use
let x;
x = 12;
// Can declare and assign at same time
let y = 10 * x;
```

Python	JavaScript
Statements	
No delimiter for the end of a simple statement A suite of statements is multiple simple statements at the same indentation level. Compound statements (function definition, if, elif, else, for) consist of a header, and a body separated by a colon. The body is a suite of statements indented one level deeper than the header.	Simple statements must end with a semi-colon ; A block of statements is multiple simple statements surrounded by curly braces { }. Compound statements (function definition, if, else, for) consist of a header and a body. The body is a block of statements.
Functions	

`# Function header followed by a body.`
`# Body is a suite of statements.`
`# Suite must be indented.`
`def foo(x, y, z):`
`  a = (x + y + z) / 3`
`  return a`
	`// Function header followed by a body.`
	`// The body is a block of statements`
	`// delimited by { }`
	`function foo(x, y, z) {`
	`  let a = (x + y + z) / 3;`
	`  return a;`
	`}`

Control Flow: Selection	

`# Starts with a single if`
`# May be followed by 0 or more elif`
`# Ends with optional else`
`if x < y:`
`  print("x is smaller")`
`elif x > y:`
`  print("y is smaller")`
`else:`
`  print("They are the same!")`
	`// Starts with a single if`
	`// May be followed by 0 or more else if`
	`// Ends with optional else`
	`if (x < y) {`
	`  console.log("x is smaller");`
	`} else if (x > y) {`
	`  console.log("y is smaller");`
	`} else {`
	`  console.log("They are the same!");`
	`}`

Python

JavaScript

Control Flow: Repetition

```python
# For loops always loop over elements
# of a sequence.

# A range is a sequence.
for i in range(0,11,3):
  print(i)

# A list is a sequence.
for x in [1, 2, 3, 4]:
  print(x)

# A string is a sequence.
for c in "Hello World!":
  print(c)
```

```javascript
// For loops can loop over elements of
// a sequence (arrays or strings).
for (let x of [1, 2, 3, 4]) {
  console.log(x);
}

// They can also loop over indices
for (let i in seq) {
  console.log(seq[i]);
}

// Or we can be explicit
for (let i=0; i<seq.length; i=i+1) {
  console.log(seq[i]);
}
```

Ordered Collections

```python
# Called Lists in Python
x = [] # Empty list
y = ["a", "b", "c"] # Non-empty list

# Add to the end of a list
x.append("hi")

# Remove item at a specific index
c1 = y.pop(1) # c1 will be "b"

# Remove the last item
c2 = y.pop() # c2 will be "c"

# Get the length
size = len(y) # size will be 1
```

```javascript
// Called Arrays in JavaScript
let a = []; // Empty array
let b = ["a", "b", "c"]; // Non-empty

// Add to the end of an array
a.push("hi");

// Remove the last item
let c = b.pop(); // c will be "c"

// Get the length
let size = b.length; // size will be 2
```

Python

JavaScript

Associative Collections

```python
# Called Dictionaries in Python
d1 = {} # Empty Dictionary
d2 = {"a":1, "b":2, "c":3} # Non-empty

# Member access is by key
d2["a"] = 3    # Update value for "a"
d2["z"] = 12   # Add "z" with value 12
print(d2["a"]) # Prints out 3

# Can make bulk updates with update()
# This updates value of "a" to 4
# and adds "d":6 as a key-value pair
d2.update({"a":4, "d":6})

# The get function also allows us to
# access values, and if the key does
# not exist we can give a default value
d2.get("a", 0) # returns 4
d2.get("x", 0) # returns 0

# del and pop() can be used to remove
# items from the dictionary
del d2["a"]   # removes "a":4 from d2
v=d2.pop("b") # removes b and returns 2

# in and not in check membership
"a" in d2 # evaluates to False
"c" in d2 # evaluates to True
"x" not in d2 # evaluates to True

# Access to keys, values, and kv pairs
# as a sequence is also possible
d2.keys()
d2.values()
d2.items()
```

```javascript
// Called Objects in JavaScript
let x = {}; // Empty Object
let y = {'a':1, 'b':2}; // Non-empty

// Member access is by key
y['a'] = 12; // Can use a string
y.b = 7;     // or a literal as the key
y['c'] = 3;  // Can add keys as well

console.log(y['c']) // prints 3
console.log(y.c)    // prints 3

// Removal uses the delete keyword
delete y['b']  // Removes 'b':7
delete y.c     // Removes 'c':3

// Membership checks use the in keyword
'c' in y    // Returns false
!('c' in y) // Returns true

// Access to keys, values, and kv pairs
// as a sequence is also possible
Object.keys(y);
Object.values(y);
Object.entries(y);
```

Python	JavaScript

Miscellaneous

```python
s# Comments start with '#'

# Output uses the print function
print("Hello World!")

# Assertions are the assert keyword
# followed by a boolean condition
# and an optional expression.
assert x < y, str(x)+">="+str(y)

# Open a file using open, and with...as
with open("file.txt", "r") as f:
  # Do something with f
  # Files are a sequence of lines
  for line in f:
    # Do something with each line
    # line is a string

# The csv library allows us to read csv
import csv
with open("f.csv","r",newline="") as f:
  # Create a csv reader
  reader = csv.reader(f)
  # Do something with file
  # CSV readers are a sequence of lines
  for line in reader:
    # Do something with each line
    # line is a list of value
```

```javascript
// Comments start with '//'
/* Comments in JavaScript can also
   span multiple lines */

// Output uses the console.log function
console.log("Hello World!");

// Assertions use the assert function,
// passing a boolean value, and an
// optional expression.
assert(x < y, x + ">=" + y);
```