	JavaScript	Python	Java	C++
absolute pathname		os.path.abspath('..')		
absolute value			Math.abs(-7) Math.abs(-7.77)	#include int i = -7; abs(i); #include float x = -7.77; fabs(x)
access control			*access keywords required for methods and members:* public class Foo { private int privateInt; protected int protectedInt; public int publicInt; }	*access keywords define regions:* class Foo { int privateInt1; int privateInt2; public: int publicInt1; int publicInt2; protected: int protectedInt1; int protectedInt2; private: int privateInt3; int privateInt4; };
add time duration		import datetime delta = datetime.timedelta(minutes=10, seconds=3) t = datetime.datetime.now() + delta		
address copy, shallow copy, deep copy		import copy a = [1,2,[3,4]] a2 = a a3 = list(a) a4 = copy.deepcopy(a)		
allocate array on heap			int[] a = new int[10];	int *a = new int[10];
allocate array on stack			*arrays must be allocated on heap*	int a[10];
allocate primitive type on heap			*primitive types are always stack allocated. Use a wrapper class to store on the heap:* Integer i = new Integer(0);	int *ip = new int;
allocate string			String s = "hello"; String t = new String(s);	string *s = new string("hello");
anonymous class			(new Object() { public void hello() { System.out.println("hello!"); } }).hello();	*possible but not useful*
anonymous function				
are expressions statements	*yes*			
arithmetic and logic				
arithmetic expression	1 + 3			
arithmetic functions	Math.sqrt Math.exp Math.log Math.sin Math.cos Math.tan Math.asin Math.acos Math.atan Math.atan2	from math import sqrt, exp, log, \ sin, cos, tan, asin, acos, atan, atan2		
arithmetic operators	+ - * / *none* % Math.pow(*base*, *exp*)	+ - * / // % **	+ - * / %	+ - * / %
arithmetic truncation	*none* Math.round(3.1) Math.ceil(3.1) Math.floor(3.1) Math.abs(-3)	import math int(x) int(round(x)) math.ceil(x) math.floor(x) abs(x)	(long)3.77 Math.round(3.77) (long)Math.floor(3.77) (long)Math.ceil(3.77)	#include double d = 3.77; long trunc = (long)d; long rnd = round(d); long flr = floorl(d); long cl = ceill(d);
array access			a[0]	a[0]
array iteration			for (String name : names) {	int a[10]; for (i=0; i<10; i++) { *do something with a[i]* }
array literal			int[] a = {1,2,3};	int a[] = {1,2,3};
array out-of-bounds result			ArrayIndexOutOfBoundsException	*undefined, possible SIGSEGV*
arrays				
arrays as function arguments		*parameter contains address copy*		
assignment	x = 1;	*assignments can be chained but otherwise don't return values:* v = 1		
backreference in match and substitution		*none* rx = re.compile('(\w+) (\w+)') rx.sub(r'\2 \1', 'do re')		
backticks	var exec = require('child_process').exec; var f = function(err, fout, ferr) { *output in fout* }; var child = exec('ls', f);	import subprocess cmd = ['ls', '-l', '/tmp'] files = subprocess.check_output(cmd)		
base conversion		*none* int("60", 7)		
binary, octal, and hex literals		0b101010 052 0x2a		
bit operators	<< >> & | ^ ~	<< >> & | ^ ~	<< >> & | ^ ~	<< >> & | ^ ~ bitand bitor comp
block delimiters	{}	*offside rule*		
boolean types			boolean	bool
break and continue	break continue			
break, continue, redo		break continue *none*		
build date/time from parts	var yr = 1999; var mo = 9; var dy = 10; var hr = 23; var mi = 30; var ss = 0; var t = new Date(yr,mo-1,dy,hr,mi,ss);			
build pathname	var path = require('path'); path.join("/etc", "hosts");	os.path.join('/etc', 'hosts')		
c-style for		*none*		
case and underscores in names			AClassName aMethodName() aVariableName	A_MACRO_NAME AClassName AMethodName() *or* a_method_name() a_variable_name

	JavaScript	Python	Java	C++
case insensitive match test	"Lorem".match(/lorem/i)	re.search('lorem', 'Lorem', re.I)		
case manipulation	"lorem".toUpperCase() "LOREM".toLowerCase() *none*	lorem'.upper() 'LOREM'.lower() 'lorem'.capitalize()		
catch exception	try { risky(); } catch (e) { alert("risky failed"); }	try: risky() except: print('risky failed')	try { throw new Exception("failed"); } catch (Exception e) { System.out.println(e.getMessage()); }	try { throw exception(); } catch (exception& e) { cout << "failed" << endl; }
catch exception by type		try: raise Bam() except Bam as e: print(e)		
character class abbreviations and anchors	*char class abbrevs:* . \d \D \s \S \w \W *anchors:* ^ $ \b \B	*char class abbrevs:* . \d \D \s \S \w \W *anchors:* ^ $ \A \b \B \Z		
character escapes		*single and double quoted:* *newline* \\ \' \" \a \b \f \n \r \t \v *ooo* \x*hh* *Python 3:* \u*hhhh* \U*hhhhhhhh*		
character translation		from string import lowercase as ins from string import maketrans outs = ins[13:] + ins[:13] 'hello'.translate(maketrans(ins,outs))		
chomp		line = line.rstrip("\r\n')		
chr and ord	String.fromCharCode(65) "A".charCodeAt(0)	chr(65) ord('A')		
class definition location			*top level, class block, or function block for anonymous classes*	*top level, class block, or function block*
class name			String name = c.getName();	typeid(Foo).name()
clone object	var o2 = Object.create(o);			
close file	fs.closeSync(f);	f.close()		
closure		# Python 3: def make_counter(): i = 0 def counter(): nonlocal i i += 1 return i return counter nays = make_counter()		
coalesce			String s1 = s2 == null ? "was null" : s2;	string s1 = s2 || "was null";
command line args	process.argv.length process.argv[0] process.argv[1] …			
command line args, script name		len(sys.argv)-1 sys.argv[1] sys.argv[2] *etc* sys.argv[0]		
command line script		$ python -c "print('hi')"		
comment out multiple lines	/* comment another comment */	*use triple quote string literal:* "'comment line another line"'		
comparison			"hello".compareTo("world")	string *s1 = new string("hello"); string *s2 = new stringt("world"); cout << s1->compare(*s2) << endl;
comparison operators	=== !== < > >= <= *perform type coercion:* == !=	*comparison operators are chainable:* == != > < >= <=		
complex numbers		z = 1 + 1.414j z.real z.imag		
compound assignment operators: arithmetic, string, logical, bit		# do not return values: += -= *= /= //= %= **= += *= &= |= ^= <<= >>= &= |= ^=		
concatenate		s = 'Hello, ' s2 = s + 'World!' *juxtaposition can be used to concatenate literals:* s2 = 'Hello, ' "World!"	"hello" + " world"	string *s1 = new string("hello"); string *s2 = new string(" world"); cout << *s1 + *s2 << endl;
concatenation	a = [1,2,3].concat([4,5,6]);			
conditional expression	x > 0 ? x : -x	x if x > 0 else -x		
constant declaration		# uppercase identifiers # constant by convention PI = 3.14		
constructor			public Rational(int n, int d) throws Exception { if (d == 0) { throw new Exception("zero denominator"); } if (d < 0) { this.num = -1 * n; this.denom = -1 * d; } else { this.num = n; this.denom = d; } }	Rational::Rational(int n, int d) : num(n), denom(d) { if (denom == 0) { throw "zero denominator"; } int div = gcd(n,d); num = num / div; denom = denom / div; }
control structure keywords		elif else for if while		
convert from string	7 + parseInt("12", 10) 73.9 + parseFloat(".037")			
convert from string, to string		7 + int('12') 73.9 + float('.037') 'value: ' + str(8)		
convert to string	"value: " + 8			
copy file, remove file, rename file	var fs = require('fs'); *??* fs.unlink("/tmp/foo"); fs.rename("/tmp/bar", "/tmp/foo");	import shutil shutil.copy('/tmp/foo', '/tmp/bar') os.remove('/tmp/foo') shutil.move('/tmp/bar', '/tmp/foo')		
create blank object	var o = new Object(); *or* var o = {};			

	JavaScript	Python	Java	C++
create object			Rational r = new Rational(7,3);	Rational r1(7,3); Rational *r2 = new Rational(8,5);
current date/time	var t = new Date();	import datetime t = datetime.datetime.now() utc = datetime.datetime.utcnow()	long millis = System. currentTimeMillis(); Date dt = new Date(millis);	
current unix epoch		import datetime t = datetime.datetime.now() epoch = int(t.strftime("%s"))		
custom delimiters		*none*		
date/time type		datetime.datetime	java.util.Date	
dates and time				
declare and access global variable	// assign without using var g = 1; function incr_global () { g++; }			
declare local variable	var x = 1;			
declare namespace			package foo.bar; public class Baz { public static final int ANSWER = 42; }	namespace foo { namespace bar { class Baz { static const int ANSWER = 42; }; } }
declare primitive type on stack			int i; int j = 3;	int i; int j = 3; int k(7);
decorator		def logcall(f): def wrapper(*a, **opts): print('calling ' + f.__name__) f(*a, **opts) print('called ' + f.__name__) return wrapper @logcall def square(x): return x * x square(5)		
dedupe		a = [1,2,2,3] a2 = list(set(a)) a = list(set(a))		
default argument value			*use method overloading*	float log(float exp, float base=10.0) {
default format example		23/08/2011 19:35:59		
default scope	*global unless declared with* var			
default value	*none*	import math def my_log(x, base=10): return math.log(x)/math.log(base) my_log(42) my_log(42, math.e)		
default value, computed value		from collections import defaultdict counts = defaultdict(lambda: 0) counts['foo'] += 1 class Factorial(dict): def __missing__(self, k): if k > 1: return k * self[k-1] else: return 1 factorial = Factorial()		
define class			public class Rational { public int num; public int denom; public Rational add(Rational o) throws Exception { return new Rational(this.num*o.denom + o.num*this.denom,this.denom*o.denom); } public static Rational max(Rational a, Rational b) { return (a.num*b.denom > a.num*b.denom) ? a : b; } }	*Rational.hpp:* class Rational { public: int num, denom; Rational(int num, int denom); virtual ~Rational(); Rational operator+(Rational& addend); static Rational max(Rational& a, Rational& b); };
define class method			*declare static in class definition*	*declare static in class definition*
define exception		class Bam(Exception): def __init__(self): super(Bam, self).__init__('bam!')		
define generic type			public class Foo { public A a; public Foo(A a) { this.a = a; } }	template class Foo { public: A a; Foo(A a); }; template Foo::Foo(A a) : a(a) { }
define method	o.doubleScore = function() { return this.score * 2; };		public int height() { return (Math.abs(this.num) > this.denom) ? Math.abs(this.num) : this.denom; }	int Rational::height() { return (abs(num) > abs(denom)) ? abs(num) : abs(denom); }
delete	delete d["t"]; delete d.t;			
delete entry		d = {1: True, 0: False} del d[1]		
destroy object			*none*	delete r2;
destructor			protected void finalize() throws Throwable { super.finalize(); }	Rational::~Rational() {};
dictionaries				
directories				
directory test		os.path.isdir('/tmp')		

	JavaScript	Python	Java	C++
dirname and basename	var path = require('path'); path.dirname("/etc/hosts"); path.basename("/etc/hosts");	os.path.dirname('/etc/hosts') os.path.basename('/etc/hosts')		
division by zero	Infinity	*raises* ZeroDivisionError	*throws* java.lang.ArithmeticException	*process sent a* SIGFPE *signal*
dynamic dispatch			*dispatch dynamic by default*	*declare as virtual in base class*
empty test		not a		
enum			public enum DayOfWeek { MON, TUE, WED, THU, FRI, SAT, SUN }; DayOfWeek d = DayOfWeek.TUE;	enum day_of_week { mon, tue, wed, thu, fri, sat, sun }; day_of_week d = tue;
environment variable			String home = System.getenv ("HOME");	#include char *home = getenv("HOME"); setenv("EDITOR", "emacs", 1); unsetenv("EDITOR");
escaped external command		import subprocess cmd = ['ls', '-l', '/tmp'] if subprocess.call(cmd): raise Exception('ls failed')		
escapes	*single and double quotes:* \b \f \n \r \t \v \u*hhhh* \x*hh* \" \' \\		\b \f \n \r \t \u*hhhh* \\ \" \' *o* *oo* *ooo*	\a \b \f \n \r \t \v \x*hh* \\ \" \' *o* *oo* *ooo*
eval	x = eval("1 + 1");			
executable test		os.access('/bin/ls', os.X_OK)		
execution control				
exit	process.exit(0)	sys.exit(0)		
external command	var exec = require('child_process').exec; var child = exec('ls');	if os.system('ls -l /tmp'): raise Exception('ls failed')		
extra arguments	*available in* arguments			
extract character		lorem ipsum'[6]		
extract substring	"lorem ipsum".substr(6, 5) "lorem ipsum".substring(6, 11)	lorem ipsum'[6:11]		
falsehoods	false null undefined "" 0 NaN	False None 0 0.0 '' [] {}	FALSE	false 0 0.0 NULL
file test, regular file test	var path = require('path'); path.existsSync("/etc/hosts");	os.path.exists('/etc/hosts') os.path.isfile('/etc/hosts')		
files				
filter	nums.filter(function(x) {return x>1})	filter(lambda x: x > 1, [1,2,3]) # or use list comprehension: [x for x in [1,2,3] if x > 1]		
finally clause			try { *risky code* } finally { *perform cleanup* }	*use local object with destructor*
finally/ensure	acquire_resource(); try { risky(); } finally { release_resource(); }	acquire_resource() try: risky() finally: release_resource()		
first argument			*first command line argument*	*pathname of executable*
float division	x / y	float(13) / 5 # Python 3: 13 / 5		
float overflow	Infinity	*raises* OverflowError		
floating point and decimal types			float *4 bytes* double *8 bytes*	float double long double
flush file handle	*none*	f.flush()		
for	for (var i=0; i<10; i++) { alert(i); }		int n = 1; for (int i=1; i<=10; i++) { n *= i; }	int i, n; for (i=1,n=1; i<=10; i++) { n *= i; }
free array on heap			*garbage collected*	delete[] a;
free primitive type on heap			*garbage collected*	delete i;
from array of pairs, from even length array		a = [[1,'a'], [2,'b'], [3,'c']] d = dict(a) a = [1,'a',2,'b',3,'c'] d = dict(zip(a[::2], a[1::2]))		
function declaration	function add(x, y) { return x+y; }	def add(a, b): return a+b		
function invocation	add(1, 2)	add(1, 2)		
function overloading			*yes*	*yes*
function reference		func = add		
function with private state		# state not private: def counter(): counter.i += 1 return counter.i counter.i = 0 print(counter())		
functions				
generator	*none*	def make_counter(): i = 0 while True: i += 1 yield i nays = make_counter() print(nays.next())		
generic array			*not permitted. Use* Object *as the element type for the array or use an*ArrayList.	template class Foo { public: C a[10]; };
generic function				template C add(C a, C b) { return a + b; }
generic type parameters				Pair > p = Pair >(7, Foo("foo"));
generic types				
get and set environment variable		os.getenv('HOME') os.environ['PATH'] = '/bin'		
get attribute	if (o.score == 21) { alert("Blackjack!"); }			

	JavaScript	Python	Java	C++
get date parts	t.getFullYear() t.getMonth() + 1 t.getDate() # getDay() is day of week			
get methods			import java.lang.reflect.*; Method[] m = c.getMethods();	
get time parts	t.getHours() t.getMinutes() t.getSeconds()			
get type class from string			Class c = Class.forName("java.io. File");	
get type class from type identifier				typeid(Foo)
get type class of object			o = new Object(); Class c = o.getClass();	
getopt		import argparse parser = argparse.ArgumentParser() parser.add_argument('--file', '-f', dest='file') args = parser.parse_args() src = args.file		
global variable		g1, g2 = 7, 8 def swap_globals(): global g1, g2 g1, g2 = g2, g1		
global variable for last exception		*last exception:* sys.exc_info()[1]		
group capture	rx = /^(\d{4})-(\d{2})-(\d{2})$/; m = rx.exec('2009-06-03'); yr = m[1]; mo = m[2]; dy = m[3];	rx = '(\d{4})-(\d{2})-(\d{2})' m = re.search(rx, '2010-06-03') yr, mo, dy = m.groups()		
has method			import java.lang.reflect.*; Class c = Class.forName("java.io. File"); Method[] a = c.getMethods(); boolean hasMethod = false; for (int i=0; i < a.length; i++) { if (a[i].getName() == "toString") { hasMethod = true; } }	
has method?	typeof(o.foo) == 'function'			
hello word			$ cat Hello.java public class Hello { public static void main(String[] args) { System.out.println("Hello, World!"); } } $ javac Hello.java $ java Hello Hello, World!	$ cat hello.cpp #include using namespace std; int main(int argc, char**arg) { cout << "Hello, World!" << endl; } $ g++ hello.cpp $ ./a.out Hello, World!
here document		*none*		
if	if (0 == n) { alert("no hits"); } else if (1 == n) { alert("1 hit"); } else { alert(n + " hits"); }	if 0 == n: print('no hits') elif 1 == n: print('one hit') else: print(str(n) + ' hits')	if (i>0) { signum = 1; } else if (i==0) { signum = 0; } else { signum = -1; }	if (i>0) { signum = 1; } else if (i==0) { signum = 0; } else { signum = -1; }
implicit prologue		import os, re, sys		
import library				
import namespace			import foo.bar.*; System.out.println(Baz.ANSWER);	using namespace foo::bar; cout << Baz::ANSWER << endl;
import part of namespace			*none*	using namespace foo; cout << bar::Baz::ANSWER << endl;
import position			*after package and before type definitions*	*anywhere a statement is legal*
import static symbol			import static foo.bar.Baz.ANSWER; System.out.println(ANSWER);	*none*
import symbol			import foo.bar.Baz; System.out.println(Baz.ANSWER);	using foo::bar::Baz; cout << Baz::ANSWER << endl;
in memory file		from StringIO import StringIO f = StringIO() f.write('lorem ipsum\n') s = f.getvalue() *Python 3 moved* StringIO *to the* io *module*		
increment and decrement		*none*		
index			"hello".indexOf("ll")	string("hello").find("ll")
index of array element		a = ['x', 'y', 'z', 'w'] i = a.index('y')		
index of substring	"lorem ipsum".indexOf("ipsum")	do re re'.index('re') 'do re re'.rindex('re') *raise* ValueError *if not found*		
indexed iteration		a = ['do', 're', 'mi', 'fa'] for i, s in enumerate(a): print('%s at index %d' % (s, i))		
inspect attributes				
inspect methods				
inspect type	typeof o			
instantiate range as array		a = range(1, 11) *Python 3:* a = list(range(1, 11))		
instatiate generic type			Foo f = new Foo("foo");	Foo f = Foo("foo");
integer division	Math.floor(x / y)			
integer division and divmod		13 // 5 q, r = divmod(13, 5)		
integer overflow	*all numbers are floats*	*becomes arbitrary length integer of type* long		
interpreter	$ node foo.js	$ python foo.py		
intersection	*none*	{1,2} & {2,3,4}		
invert		to_num = {'t':1, 'f':0} # dict comprehensions added in 2.7: to_let = {v:k for k, v in to_num.items()}		
invoke class method				
invoke method	alert("Answer: " + o.doubleScore());		r.height();	r1.height(); r2->height();

	JavaScript	Python	Java	C++
invoke method object			import java.lang.reflect.*; Class c = Class.forName("java.io. File"); Method m = c.getMethod("toString"); Object o = new Object(); m.invoke(o);	
invoking superclass constructor			super(n, 1);	Integer::Integer(int n) : Rational(n, 1) { }
is key present	d.hasOwnProperty("t");	y' in d		
iterate over a file by line	var fs = require('fs'); var file = fs.readFileSync("/etc/hosts"). toString(); file.split("\n").forEach(function (s) { *use s* });			
iterate over directory by file	var fs = require('fs'); var sys = require('sys'); var a = fs.readdirSync("/etc"); for (var i=0; i sys.puts(a[i]); }	for filename in os.listdir('/etc'): print(filename)		
iterate over file by line		for line in f:		
iterate over range		range *replaces* xrange *in Python 3:* for i in xrange(1, 1000001): *code*		
iterate thru environment variables			import java.util.Map; Map env = System.getenv(); for (String name : env.keySet()) { String value = env.get(name)); }	
iteration	var len = nums.length; for (var i=0; i alert(nums[i]); }	for i in [1,2,3]: print(i)		
join	["do", "re", "mi"].join(" ")	' '.join(['do', 're', 'mi', 'fa'])		
keys and values as arrays		d.keys() d.values() *Python 3:* list(d.keys()) list(d.values())		
lambda declaration	sqr = function(x) { return x*x; }	*body must be an expression:* sqr = lambda x: x * x		
lambda invocation	sqr(2)	sqr(2)		
length	"lorem".length	len('lorem')	s.length()	s->length()
libraries and modules				
libraries and namespaces				
libraries used			*Java API*	*STL and Boost*
library	$ cat foo.js function add(x,y) { return x+y; }			
library path	*node.js, not available in repl:* require.paths			
library path environment variable	*none*			
list installed packaged, install a package	$ npm ls $ npm install tmp			
literal	nums = [1,2,3,4]	a = [1, 2, 3, 4]	"don't say\"no\""	*none*
literal, custom delimited literal		re.compile('lorem|ipsum') *none*		
local timezone		*a* datetime *object has no timezone information unless a* tzinfo *object is provided when it is created*		
local variable declarations		# in function body: v = None a, d = [], {} x = 1 y, z = 2, 3		
logical operators	&& || !	and or not	&& || !	&& || ! and or not
lookup	nums[0]	a[0]		
lowercase			"HELLO".toLowerCase()	#include string s("HELLO"); boost::to_upper(s);
make directory	var fs = require('fs'); fs.mkdirSync("/tmp/foo", 0755); fs.mkdirSync("/tmp/foo/bar", 0755);	dirname = '/tmp/foo/bar' if not os.path.isdir(dirname): os.makedirs(dirname)		
manipulate back		a = [6,7,8] a.append(9) a.pop()		
manipulate back of array	a = [6,7,8]; a.push(9); i = a.pop();			
manipulate front		a = [6,7,8] a.insert(0,5) a.pop(0)		
manipulate front of array	a = [6,7,8]; a.unshift(5); i = a.shift();			
map	nums.map(function(x) {return x*x})	map(lambda x: x * x, [1,2,3]) # or use list comprehension: [x*x for x in [1,2,3]]		
map access			m.put("hello", 5); m.get("hello")	m["hello"] = 5; cout << m["hello"]) << endl;
map declaration			java.util.TreeMap m = new java.util. TreeMap();	#include map m;
map element not found result			null	NULL
map iterate			for (java.util.Map.Entry e : m. entrySet()) { *use e.getKey() or e.getValue()* }	map::iterator mi; for (mi = m.begin(); mi != m.end(); mi++) { printf("%s %d", mi->first, mi->second) }
map remove element			m.remove("hello");	m.erase(m.find("hello"));
map size			m.size()	m.size()
mark class underivable or method unoverrideable			final	*none*
match test	if (s.match(/1999/)) { alert("party!"); }	if re.search('1999', s): print('party!')		

	JavaScript	Python	Java	C++
match, prematch, postmatch		m = re.search('\d{4}', s) if m: match = m.group() prematch = s[0:m.start(0)] postmatch = s[m.end(0):len(s)]		
member, not a member	*none*			
membership		7 in a		
merge		d1 = {'a':1, 'b':2} d2 = {'b':3, 'c':4} d1.update(d2)		
message passing	o["foo"](1,1)			
methods must declare exceptions			*yes*	*no*
microseconds		t.microsecond		
min and max	Math.min(1,2,3) Math.max(1,2,3) Math.min.apply(Math, [1,2,3]) Math.max.apply(Math, [1,2,3])	min(1,2,3) max(1,2,3) min([1,2,3]) max([1,2,3])		
missing argument behavior		*raises* TypeError		
missing argument value	undefined			
modifiers	g i m	re.I re.M re.S re.X		
module declaration				
module separator				
multiple namespaces per file			*no*	*yes*
multiple return values	*none*	def first_and_second(a): return a[0], a[1] x, y = first_and_second([1,2,3])		
multiple type parameters				template class Pair { public: A a; B b; Pair(A a, B b); }; template Pair::Pair(A a, B b) : a(a), b(b) { } Pair p = Pair(7, "foo");
name of receiver			this	this
named parameters		def fequal(x, y, **opts): eps = opts.get('eps') or 0.01 return abs(x - y) < eps fequal(1.0, 1.001) fequal(1.0, 1.001, eps=0.1**10)	*none*	*none*
namespaces map to directories			*yes*	*no*
nested function visibility	*not visible outside containing function*			
newline in literal	*yes*	*triple quote literals only*		
newline in literal?			*no*	*string literals can extend over multiple lines, but the newlines do not appear in the resulting string*
null	null	None	null	NULL
null test	v === null	v == None v is None		
number to string			Integer.toString(14) Long.toString(14) Double.toString(14.7)	
object literal	var o = { score: 21, doubleScore: function() { return this.score * 2; } };			
objects				
open file	var fs = require('fs'); f = fs.openSync("/tmp/foo", "r");	f = open('/etc/hosts')		
open file for append		with open('/tmp/test') as f: f.write('lorem ipsum\n')		
open file for writing	var fs = require('fs'); f = fs.openSync("/tmp/foo", "w");	f = open('/tmp/test', 'w')		
operator overloading			*none*	Rational Rational::operator+(Rational& o) { return Rational(this->num*o.denom + o.num*this->denom, this->denom * o.denom); }
out of bounds behavior	*returns* undefined			
out-of-bounds behavior		a = [] *raises* IndexError: a[10] *raises* IndexError: a[10] = 'lorem'		
pad on right				
pad on right, on left		lorem'.ljust(10) 'lorem'.rjust(10)		
pad on right, pad on left, center	*none*			
pair				pair p(7, 3.14); cout << p.first << ", " << p.second << endl;
parallel assignment	*none*	x, y, z = 1, 2, 3 # raises ValueError: x, y = 1, 2, 3 # raises ValueError: x, y, z = 1, 2		
parse date w/o format	var t = new Date("July 7, 1999");	# pip install python-dateutil import dateutil.parser s = 'July 7, 1999' t = dateutil.parser.parse(s)		
pass array or dictionary by reference		def foo(x, y): x[2] = 5 y['f'] = -1 a = [1,2,3] d = {'t':1, 'f':0} foo(a, d)		

	JavaScript	Python	Java	C++
pass by address			*none*	void use_iptr(int *i) { *function body* } int i = 7; use_iptr(&i);
pass by reference			*objects and arrays are always passed by reference*	void use_iref(int& i) { printf("using iref: %d", i); } int i = 7; use_iref(i);
pass by value			*primitive types are always passed by value*	void use_integer(int i) { *function body* } int i = 7; use_integer(i);
pass number or string by reference		*not possible*		
passing functions				
power			Math.pow(2.0,3.0);	#include boost::math::powm1(2.0,3.0)+1
primitive types				
print to standard output	var sys = require('sys'); sys.puts("Hello, World!");	print('Hello, World!')		
printf			System.out.printf("count: %d", 7);	cout << "count: " << 7 << endl;
processes and environment				
quote words		*none*		
raise exception	throw "bad arg";	raise Exception('bad arg')		
random integer			java.util.Random r = new java.util. Random(); int i = r.nextInt();	#include using namespace boost; mt19937 rng; uniform_int<> ui(0,RAND_MAX); variate_generator > brand(rng, ui); int i = brand()
random integer, uniform float, normal float	Math.floor(Math.random() * 100) Math.random() *??*	import random random.randint(0,99) random.random() random.gauss(0,1)		
rational numbers		from fractions import Fraction x = Fraction(22,7) x.numerator x.denominator		
read entire file into array or string		a = f.readlines() s = f.read()		
read file	var fs = require('fs'); fs.readFileSync("/tmp/foo", "utf8");			
read from file			import java.io.BufferedReader; import java.io.FileReader; BufferedReader in = new BufferedReader(new FileReader ("/etc/passwd")); String line; while ((line = in.readLine()) != null) { *process line* }	#include string line; ifstream f("/etc/passwd"); if (f.is_open()) { while (!f.eof()) { getline(f, line); *process line* } f.close(); if (0 != f.fail()) { *handle error* } } else { *handle error* }
read from standard input		line = sys.stdin.readline()		
read line		f.readline()		
recursive copy		import shutil shutil.copytree('/tmp/foodir', '/tmp/bardir')		
recursive regex		*none*		
reduce	nums.reduce(function(m,o) { return m+o; }, 0)	# import needed in Python 3 only from functools import reduce reduce(lambda x, y: x+y, [1,2,3], 0)		
reflection				
regex match			boolean isMatch = "hello".matches(".*ll. *");	#include using namespace boost::xpressive; sregex re = sregex::compile(".*ll.*"); smatch matches; string s("hello"); bool is_match = regex_match(s, matches, re);
regex substitute			String s1 = "hello".replace("ll","LL"); String s2 = "hello".replaceAll("l","L");	#include using namespace boost::xpressive; string s("hello"); sregex re1 = as_xpr("ll"); string format1("LL"); string result1 = regex_replace(s, re1, format1, regex_constants::format_first_only); sregex re2 = as_xpr("l"); string format2("L"); string result2 = regex_replace(s, re2, format2);
regexes				
regions which define local scope		*nestable (read only): function or method body*		
regular expresions				
regular expressions				
relational expression	x > 3			
relational operators			== != < > <= >=	== != < > <= >=
relative complement, symmetric difference		{1,2,3} - {2} {1,2} ^ {2,3,4}		
remove directory and contents		import shutil shutil.rmtree('/tmp/foodir')		
remove empty directory	var fs = require('fs'); fs.rmdirSync("/tmp/foo/bar");	os.rmdir('/tmp/foodir')		
repl	$ node	$ python		
replicate		hbar = '-' * 80		

	JavaScript	Python	Java	C++
result of date subtraction		datetime.timedelta *object*		
return value	return *arg or* undefined. *If invoked with* new *and return value not an object, returns* this	return *arg or* None		
reverse	var a = [1,2,3]; a.reverse();	a = [1,2,3] a[::-1] a.reverse()		
root class			java.lang.Object	*none*
root class methods			clone() equals() finalize() getClass() hashCode() toString()	*none*
scan		s = 'dolor sit amet' a = re.findall('\w+', s)		
semantics of ==			*object identity comparison*	*value comparison*
set attribute	o.score = 21;			
set difference	*none*			
set file permissions	var fs = require('fs'); fs.chmod("/tmp/foo", 0755);	os.chmod('/tmp/foo', 0755)		
set random seed, get and restore seed		import random random.seed(17) sd = random.getstate() random.setstate(sd)		
set signal handller		import signal def handler(signo, frame): print('exiting…') exit -1 signal.signal(signal.SIGINT, handler)		
show version	$ node —version	$ python -V		
shuffle and sample		from random import shuffle, sample a = [1, 2, 3, 4] shuffle(a) sample(a, 2)		
signature of main			public class *Foo* { public static void main(String[] args) {	int main(int argc, char **argv) {
signed integer types			byte *1 byte* short *2 bytes* int *4 bytes* long *8 bytes*	signed char *1+ byte* short int *2+ bytes* int *2+ bytes* long int *4+ bytes* long long int *4+ bytes*
size	nums.length	len(a)		
sleep	*none*	import time time.sleep(0.5)		
slice	nums.slice(1,3)			
slice by endpoints, by length		*select 3rd and 4th elements:* a[2:4] *none*		
slice to end		a[1:]		
sort	var a = [3,1,4,2]; a.sort();	a = ['b', 'A', 'a', 'B'] sorted(a) a.sort() a.sort(key=str.lower)		
source, header, object file suffix			.java *none* .class	.cpp .hpp .o
split	"do re mi".split(" ")		"Bob Ned Amy".split(" ")	#include #include string s("Bob Amy Ned"); vector vec; boost::split(vec, s, boost::is_any_of(" "));
split, in two, with delimiters, into characters		do re mi fa'.split() 'do re mi fa'.split(None, 1) re.split('(\s+)', 'do re mi fa') list('abcd')		
sprintf	*none*	lorem %s %d %f' % ('ipsum', 13, 3.7) fmt = 'lorem {0} {1} {2}' fmt.format('ipsum', 13, 3.7)	String.format("%s: %d", "Spain", 7)	#include ostringstream o(''); o << "Spain" << ": " << 7; o.str();
sqrt -2	NaN	# raises ValueError: import math math.sqrt(-2) # returns complex float: import cmath cmath.sqrt(-2)		
standard file handles		sys.stdin sys.stdout sys.stderr		
start thread		class sleep10(threading.Thread): def run(self): time.sleep(10) thr = sleep10() thr.start()		
statement modifiers		*none*		
statement separator	; *or newline* *newline not separator inside (), [], {}, "", '', or after binary operator* *newline sometimes not separator when following line would not parse as a valid statement*	*newline or* ; *newlines not separators inside (), [], {}, triple quote literals, or after backslash: *		
static dispatch			*declare as final, private, or static (i.e. make it a class method)*	*dispatch static by default*
strftime	*none*	t.strftime('%Y-%m-%d %H:%M:%S')	String s = "yyyy-MM-dd HH:mm:ss"; DateFormat fmt = new SimpleDateFormat(s); String s2 = fmt.format(dt);	
string concatenation	s = "Hello, " + "World!";			
string literal	"don't say \"no\"" 'don\'t say "no"'	don\'t say "no"' "don't say \"no\"" "don't " 'say "no"' "'don't say "no"'" """don't say "no\"""""		

	JavaScript	Python	Java	C++
string to number			Byte.parseByte("14") Short.parseShort("14") Integer.parseInt("14") Long.parseLong("14") Float.parseFloat("14.7") Double.parseDouble("14.7")	#include stringstream ss("7 14.3 12"); int i; double d; long l; ss >> i >> d >> l;
strings				
strip	" lorem ".trim() # some browsers: " lorem".trimLeft() "lorem ".trimRight()	lorem '.strip() ' lorem'.lstrip() 'lorem '.rstrip()		
strptime	*none*	from datetime import datetime s = '2011-05-03 10:00:00' fmt = '%Y-%m-%d %H:%M:%S' t = datetime.strptime(s, fmt)	String s = "2011-05-03 17:00:00"; Date dt2 = fmt.parse(s);	
struct declaration			MedalCount spain = new MedalCount();	MedalCount spain;
struct definition			public class MedalCount { public String country; public int gold; public int silver; public int bronze; }	class MedalCount { public: const char *country; int gold; int silver; int bronze; };
struct initialization			*no object literal syntax; define a constructor*	MedalCount spain = { "Spain", 3, 7, 4 };
struct member access			int spain_total = spain.gold + spain. silver + spain.bronze;	int spain_total = spain.gold + spain. silver + spain.bronze;
struct member assignment			spain.country = "Spain"; spain.gold = 3; spain.silver = 7; spain.bronze = 4;	spain.country = "Spain"; spain.gold = 3; spain.silver = 7; spain.bronze = 4;
subclass			public class RInteger extends Rational { public RInteger(int n) throws Throwable { super(n, 1); } }	class Integer : public Rational { public: Integer(int n); virtual ~Integer(); };
substitution	s = "do re mi mi mi"; s.replace(/mi/g, "ma");	s = 'do re mi mi mi' s = re.compile('mi').sub('ma', s)		
substring			"hello".substring(2,4)	string("hello").substr(2,2)
swap	tmp = x; x = y; y = tmp;	x, y = y, x		
switch		*none*	switch(i) { case 0: 0; break; case 1: 1; break; default: -1; break; }	switch(i) { case 0: 0; break; case 1: 1; break; default: -1; break; }
template parameter				
template parameters				
template specialization				
temporary file		import tempfile f = tempfile.NamedTemporaryFile(prefix='foo') f.write('lorem ipsum\n') f.close() print("tmp file: %s" % f.name)		
three value comparison		*removed from Python 3:* cmp(0, 1) cmp('do', 're')		
throw exception			throw new Exception("failed");	throw exception();
timeout		import signal, time class Timeout(Exception): pass def timeout_handler(signo, fm): raise Timeout() signal.signal(signal.SIGALRM, timeout_handler) try: signal.alarm(5) time.sleep(10) except Timeout: pass signal.alarm(0)		
timezone name; offset from UTC; is daylight savings?		import time tm = time.localtime() time.tzname[tm.tm_isdst] (time.timezone / -3600) + tm.tm_isdst tm.tm_isdst		
to C string			*none*	s->c_str()
to unix epoch, from unix epoch	Math.round(t.getTime() / 1000) var epoch = 1315716177; var t2 = new Date(epoch * 1000);	from datetime import datetime as dt epoch = int(t.strftime("%s")) t2 = dt.fromtimestamp(1304442000)	long epoch = dt.getTimeInMillis()/1000; Date dt2 = new Date(epoch * 1000);	
to-end-of-line comment	// comment	# comment		
transcendental functions			Math.sqrt Math.exp Math.log *none* Math.log10 Math.sin Math.cos Math. tan Math.asin Math.acos Math.atan Math. atan2	#include sqrt exp log log2 log10 sin cos tan asin acos atan atan2
trim			" hello ".trim()	#include string s(" hello "); boost::trim(s);
true and false	true false	True False	true false	true false
type			java.lang.String	std::string
typedef			*none*	typedef int customer_id; customer_id cid = 3;

	JavaScript	Python	Java	C++
uncaught exception behavior	*error to console; script terminates. Other scripts in page will execute*			
undefined test	v === undefined	not_defined = False try: v except NameError: not_defined = True		
undefined variable access	undefined	*raises* NameError		
union	*none*	{1,2} | {2,3,4}		
universal and existential tests		all(i%2 == 0 for i in [1,2,3,4]) any(i%2 == 0 for i in [1,2,3,4])		
unsigned integer types			char *2 bytes*	unsigned char: 8+ unsigned short int *2 bytes+* unsigned int *2 bytes+* unsigned long int *4+ bytes* unsigned long long int *4+ bytes*
update	d["t"] = 2; d.t = 2;	a[0] = 'lorem'		
uppercase			"hello".toUpperCase()	#include string s("hello"); boost::to_upper(s);
url encode/decode			import java.net.URLEncoder; import java.net.URLDecoder; String url = "http://www.google.com"; String s = URLEncoder.encode(url, "utf8"); String s2 = URLDecoder.decode(s, "utf8");	
using a symbol that hasn't been imported			System.out.println(foo.bar.Baz.ANSWER);	cout << foo::bar::Baz::ANSWER << endl;
value of uninitialized primitive types			*zero-initialized*	*same as C. However, C++ provides a no-argument constructor for each primitive type which zero-initializes it.*
value parameter				template int add(int i) { return N+i; } cout << add<7>(3) << endl;
variable interpolation	*none*	count = 3 item = 'ball' print('{count} {item}s'.format(**locals()))		
variable number of arguments	*args in* arguments[0], arguments[1], … *with number of args in*arguments.length	def foo(*a): if len(a) >= 1: print('first: ' + str(a[0])) if len(a) >= 2: print('last: ' + str(a[-1]))	public static String concat(String first, String… rest) { StringBuilder sb = new StringBuilder (first); for (String arg: rest) { sb.append(arg); } return sb.toString(); } String s = Concat.concat("Hello", ", ", "World", "!");	*use C; use default values or function overloading for finite number of arities*
vector access			vec.elementAt(0)	vec[0] vec.at(0)
vector declaration			java.util.Vector vec = new java.util.Vector();	#include vector vec;
vector iteration			for (String s : vec) { *do something with s* }	int sum = 0; vector::iterator vi; for (vi = vec.begin(); vi != vec.end(); vi++) { sum += *vi; }
vector out of bounds result			*throws* ArrayIndexOutOfBoundsException	*vec[] has undefined behavior* *vec.at() raises* out_of_range
vector pop			vec.removeElementAt(vec.size()-1);	vec.pop_back();
vector push			vec.add("hello"); *or* vec.add(vec.size(), "hello")	vec.push_back(7);
vector size			vec.size()	vec.size()
version			$ javac -version	$ g++ —version
version used	*ECMAScript 5* *node.js 0.4*		*java 1.6*	*g++ 4.2*
versions used		*2.7; 3.2*		
wait on thread		thr.join()		
web				
what do does		*raises* NameError *unless a value was assigned to it*		
while	while (i < 100) { i += 1; }	while i < 100: i += 1	int i = 0; while (i<10) { … i++; }	|int i = 0; while (i<10) { … i++; }
write to file	fs.writeSync(f, "lorem ipsum");	f.write('lorem ipsum')	import java.io.BufferedWriter; import java.io.FileWriter; BufferedWriter fout = new BufferedWriter(new FileWriter ("/tmp/test2")); int i; for (i=0; i<10; i++) { fout.write(String.format("%d", i)); fout.newLine(); } fout.close();	#include ofstream f("/tmp/test4"); int i; for (i=0; i<10; i++) { f << i << endl; } f.close(); if (0 != f.fail()) { *handle error* }
zip		# array of 3 pairs: a = zip([1,2,3], ['a', 'b', 'c'])		