	Python	Ruby	Java
Flow of Control			

Conditionals	`if (v > 10):`
	`  print "V is too large"`
		`if v > 10`
		`  puts "V is too large"`
		`end`
			`if (v > 10) {`
			`  System.out.println("V is too large");`
			`}`

While loop	`while a <= 100:`
	`  a += 1`
		`while a <= 100`
		`  a += 1`
		`end`
			`while (a <= 100) {`
			`  a++;`
			`}`

Repeat/until	`while True:`
	`  some_action()`
	`  if conditional():`
	`  break`
		`repeat`
		`  some_action()`
		`until conditional()`
			`repeat {`
			`  some_action();`
			`} until (conditional());`

Strings			

String comparison	`if a == b or c > d or e < f:`
	`  print "yes!"`
		`if a == b || c > d || e < f`
		`  puts "yes!"`
		`end`
			`if (a.equals(b) || c.compareTo(d) > 0 || e.compareTo(f) < 0) {`
			`  System.out.println("yes!");`
			`}`

String length	`print len(s)`	`puts s.length`	`System.out.println(s.length());`
String concatenation	`print "hello" + "there"`	`puts "hello" + "there"`	`System.out.println("hello" + "there");`
String interpolation	`print "The value of the two text fields are %s and %s" % (t1, t2)`	`puts "The value of the two text fields are {t1} and {t2}"`	`System.out.println(String.format("The value of the two text fields are %s and %s", t1, t2));  // Java 5+`

Removing characters from a string	`ns = s.replace("$", "")  # one character`
	`ns = re.sub(r"[,$]", "", s) # multiple characters w/RE`
		`ns = s.gsub("$", "")`
		`ns = s.gsub(/[,$]/, "")`
			`String ns = s.replace("$", ""); // one character`
			`String ns = s.replaceAll("[,\$]", "");  // multiple characters`

Splitting a string into words	`words = s.split(' ')`	`words = s.split(' ')`	`String [] words = s.split(" ");`
See if string starts/ends with something	`s.startswith("$")` `s.endswith("!")`	`s.start_with?("$")` `s.end_with?("!")`	`s.startsWith("$");` `s.endsWith("!");`
Lists / arrays			

Print out a list	`print the_list`	`puts the_list`	`for (int ix = 0; ix < the_list.size(); ix++) {`
			`  System.out.println(the_list.get(ix));`
			`}`
			
			`// or...`
			
			`the_list.forEach(System.out::println);  // Java 8+ only`

Get the length of a list	`len(the_list)`	`the_list.size` `the_list.length  // also works`	`s.length  // for arrays -- i.e., String []` `s.size()  // for Lists -- i.e., ArrayList`

Loop integers from 1 to 10, inclusive	`for i in xrange(1, 11):`
	`  my_function(i)`
		`for i = 1..10 do`
		`  my_function(i)`
		`end`
			`for (int i = 0; i <= 10; i++) {`
			`  my_function(i);`
			`}`

Do something for every element in a list	for e in elements:
	  if e.text() == "Sale":
	  sale = True
		for e in elements
		  if e.text == "Sale"
		  sale = true
		  end
		end
		
		// Or...
		
		elements.each do |e|
		  if e.text == "Sale"
		  sale = true
		  end
		end
			for (WebElement e: elements) {
			  if (e.text().equals("Sale")) {
			  sale = true;
			  }
			}

Create a new list from an old list by doing something to each element	tv = [e.text() for e in elements]	tv = elements.map { |e| e.text }	List<String> tv = new ArrayList<String>();
			for (WebElement e: elements) {
			  tv.add(e.text());
			}
			
			List<String> b_list =
			a_list.stream().map(WebElement::text).collect(Collectors.toList());
			  // Java 8+ only

See if a list has any elements	elements =
	driver.find_elements(By.XPATH,
	'//button')
	if elements:
	  print "Elements found"
		elements =
		driver.find_elements(:xpath
		=> "//button")
		if elements.count > 0
		  print "Elements found"
		end
			List<WebElement> elements =
			driver.findElements(By.xpath("//button"));
			if (elements.size() > 0) {
			  System.out.println("Elements found");
			}

Sort a list	sorted_list = sorted(original_list) sorted_list = sorted(original_list, reverse=True) # reverse sort	sorted_list = original_list.sort() sorted_list = original_list.sort().reverse() sorted_list = original_list.sort{ |a,b| b <=> a } # alternate reverse sort	List sorted_list = new ArrayList(original_list); Collections.sort(sorted_list); Collections.reverse(sorted_list); // for reverse sort
Dictionaries / Hash Maps			

Initialize a hash map	the_map = { 1 : "one", 2 : "two"
	}
		the_map = { 1 => "one", 2
		=> "two" }
			// oh Java, you make me cry
			HashMap<Integer, String> the_map = new HashMap<Integer,
			String>() {{
			  put(1, "one");
			  put(2, "two");
			}};

Getting / setting an element	the_map[key] the_map[key2] = new_value	the_map[key] the_map[key2] = new_value	the_map.get(key) the_map.put(key2, new_value)

See if an element exists	if key in the_map:
	  print "key exists"
		if the_map.has_key?(key)
		  puts "key exists"
		end
			if (the_map.containsKey(key)) {
			  System.out.println("key exists");
			}

Iterate through a hash map by key	for key in the_map:
	  # do stuff here with
	the_map[key]
		for key in map do
		  # do stuff here with
		the_map[key]
		end
			for (String s: the_map.keySet()) {
			  // do stuff here with the_map.get(key)
			}

Functions and Exceptions			

Defining functions	def foo(a, b):
	  return a+b
		def foo(a, b)
		  return a+b
		end
			function foo(int a, int b) {
			  return a+b;
			}

	foo(10,20) # two parameters
	foo(10) # one parameter
	foo() # zero parameters
		foo(10,20) # two
		parameters
		foo(10) # one
		parameters
			foo(10,20);
			foo(10);
			foo();

Calling functions		foo 10 # also allowed foo() # zero parameters foo # also allowed	
Throwing Exceptions	raise Exception("Boom")	raise "Boom"	throw new Exception("Boom");

Catching Exceptions	try:
	  something_risky()
	except Exception, e:
	  print "Caught Exception!"
		begin
		  something_risky()
		rescue
		  print "Caught Exception!"
		end
			try {
			  something_risky();
			} catch (Exception e) {
			  System.out.println("Caught Exception!");
			}