Python 3	Java
General	
`# this is a comment`	`// this is a comment`
`# no special character at end of line`	`// semicolon required at end of line`

`# no types when declaring variables`
`name = "Lee"`
`age = 38`
	`// type required when declaring variables`
	`String name = "Lee";`
	`int age = 38;`

`print("Hello, world!")`	`System.out.println("Hello, world!");`
Arithmetic	
`a = 1` `b = 7.89`	`int a = 1;` `double b = 7.89;`
`2 + 3  # returns 5`	`2 + 3  // returns 5`

`7 / 2  # returns 3.5`
`7 // 2  # returns 3`
`7 % 2  # returns 1`
	`7.0 / 2.0  // returns 3.5`
	`7 / 2  // returns 3`
	`7 % 2  // returns 1`

`x += 10  # adds 10 to x`	`x += 10  // adds 10 to x`
`pow(5,2)  # returns 25` `5 ** 2  # returns 25`	`Math.pow(5,2)  // returns 25`

`import math`
`math.sqrt(25)  # returns 5`
`25 ** 0.5  # returns 5`
	
	`Math.sqrt(25)  // returns 5`
	`Math.pow(25, 0.5)  // returns 5`

`abs(-3)  # returns 3`	`Math.abs(-3)  // returns 3`

`round(4.3)  # returns 4`
`round(4.5)  # returns 4`
`round(4.7)  # returns 5`
	`Math.round(4.3)  // returns 4`
	`Math.round(4.5)  // returns 4`
	`Math.round(4.7)  // returns 5`

`min(8,9)  # returns 8` `max(8,9)  # returns 9`	`Math.min(8,9)  # returns 8` `Math.max(8,9)  # returns 9`
`import random` `x = random.random()  # 0.0 <= x < 1.0`	 `double x = Math.random();  // 0.0 <= x < 1.0`
Text	
`word = "abcdefg"`	`String word = "abcdefg";`
`len(word)  # returns 7`	`word.length();  // returns 7`
`word[2]  # returns 'c'`	`word.charAt(2);  // returns 'c'`
`word[2:5]  # returns 'cde'`	`word.substring(2,5);  // returns "cde"`
`word.upper()  # returns 'ABCDEFG'`	`word.toUpperCase();  // returns "ABCDEFG"`
`"num" + "ber"  # returns 'number'`	`"num" + "ber";  // returns "number"`

`"num" + str(337)  # returns 'num337'`	`"num" + 337;          // returns "num337"`

`# getting text input`
`name = input("Enter your name: ")`
	`// getting text input`
	`import java.util.Scanner;`
	`Scanner scan = new Scanner(System.in);`
	`System.out.print("Enter your name: ");`
	`String name = scan.next();`

`letters = ["a", "b", "c"]` `",".join(stuff)      # returns "a,b,c"`	`String[] letters = {"a", "b", "c"};` `String.join(",", letters) // returns "a,b,c"`
`"d,e,f".split(",")   # returns ["d","e","f"]`	`"d,e,f".split(",")        // returns ["d","e","f"]`
Conversion	
`str(337)       # returns "337"`	`Integer.toString(337)       // returns "337"` `"" + 337                    // returns "337"`
`int("337")     # returns 337`	`Integer.parseInt("337")     // returns 337`
`float("3.14")  # returns 3.14`	`Float.parseFloat("3.14")    // returns 3.14f` `Double.parseDouble("3.14")  // returns 3.14`
Booleans and Conditionals	
`# values: True, False`	`// values: true, false`
`# comparisons: ==, <, <=, >, >=, !=`	`// comparisons: ==, <, <=, >, >=, !=`
`# operators: and, or, not`	`// operators: &&, ||, !`

`if (num < 0):              # note the colon`
`    print("negative")     # group with spaces`
`    print("not positive") #   not braces`
`elif (num == 0):`
`    print("zero")`
`else:`
`    print("positive")`
	`if (num < 0)     // note: no colon after condition`
	`{                 // group with braces, not spaces`
	`    System.out.println("negative");`
	`    System.out.println("negative");`
	`}`
	`else if (num == 0)`
	`    System.out.println("zero");`
	`else`
	`    System.out.println("positive");`

Collections	

`# elements can be different types`
 	
`# tuple: size/elements can not be modified`
`# list:  size/elements can be modified`
	`// elements are the same type`
	`//  (an "Object" collection could have different)`
	`// array: elements can be modified (but not size)`
	`// list:  size/elements can be modified`

`# tuple`
`tuple = ("a", "b", "c", "z")`
`len(tuple)     # returns 4`
`tuple[2]       # returns "c"`
	`// array`
	`String[] array = {"a", "b", "c", "z"};`
	`array.length      // returns 4`
	`array[2]          // returns "c"`

# list	// list

list = [42, 14, 337]	import java.util.ArrayList;
	ArrayList list = new ArrayList();
	// option 1:
	list.add(42); list.add(14); list.add(337);
	// option 2:
	import java.util.Collections;
	Collections.addAll(list, 42, 14, 337);
	// option 3:
	import java.util.Arrays;
	list = new ArrayList(Arrays.asList(42, 15, 337));

list[1] # returns 14	list.get(1) // returns 14
len(list) # returns 3	list.size() // returns 3
list.append(25) # returns [42, 14, 337, 25]	list.add(25) // returns [42, 14, 337, 25]

# remove element from position 	
del list[2] # returns [42, 14]
# remove element with value 	
list.remove(14) # returns [42, 337]
	// remove element from position
	list.remove(2)
	// remove element with value; cast if integer type
	list.remove((Integer)14)

14 in list # returns True
15 in list # returns False
15 not in list # returns True
	list.contains(14) // returns true
	list.contains(15) // returns false
	!list.contains(15) // returns true

list.index(337) # returns 2	list.indexOf(337) // returns 2
list[1:3] # returns [14, 337]	list.subList(1,3) // returns [14, 337]

min(list) # returns 14
max(list) # returns 337
	Collections.min(list) // returns 14
	Collections.max(list) // returns 337

list.sort() # returns [14, 42, 337]	Collections.sort(list) // returns [14, 42, 337]
import random random.shuffle(list) # shuffles elements	Collections.shuffle(list) // shuffles elements
list + [8,9] # returns [14, 42, 337, 8, 9]	list.addAll(Arrays.asList(8,9)); // returns [14, 42, 337, 8, 9]
list(range(4)) # returns [0, 1, 2, 3] list(range(3,7)) # returns [3, 4, 5, 6] list(range(0,10,2)) # returns [0, 2, 4, 6, 8]	

Loops

```
n = 0
while n < 10:
    print("Value " + str(n))
    n += 1
```

```
int n = 0;
while (n < 10)
{
    System.out.println("Value " + n);
    n += 1;
}
```

```
for n in range(10):
    print("Value " + str(n))
```

```
for (int n = 0; n < 10; n++)
    System.out.println("Value " + n);
```

```
for name in ["Alice", "Bob", "Cate"]:
    print("Hello, " + name)
```

```
String[] list = {"Alice", "Bob", "Cate"};
for (String name: list)
    System.out.println("Hello, " + name);
```

Dictionary/Mapping

```
dict = {"name":"Lee", "age":37}
```

```
HashMap map = new Hashmap();
map.put("name","Lee");  map.put("age",37);
```

`len(dict)`	`# returns 2`	`map.size()`	`// returns 2`
`dict["name"]`	`# returns "Lee"`	`map.get("name")`	`// returns "Lee"`
`dict["age"] = 38`	`# updates value`	`map.replace("age", 38)`	`// updates value`
`dict["teacher"] = True`	`# adds key/value`	`map.put("teacher", true)`	`// adds key/value`
`del dict["age"]`	`# removes key/value`	`map.remove("age")`	`// removes key/value`
`"name"  in dict` `"stuff" in dict`	`# returns True` `# returns False`	`map.containsKey("name")` `map.containsKey("stuff")`	`// returns true` `// returns false`
`list(dict.keys())`	`# ["name", "age"]`	`map.keySet()`	`// ["name", "age"]`
`list(dict.values())`	`# ["Lee", 37]`	`map.values()`	`// ["Lee", 37]`

Functions

```
# define
def average(x,y):
     return (x+y)/2


# usage
average(3,7)
```

```
# define - all functions must be in a class
class MyMath
{
   static double average(double x, double y)
   {  return (x+y)/2;  }
}

# usage
MyMath.average(3,7)
```

Classes

```
class Person(object):

  # parameters with default values
  def __init__(self, n="Lee"):
    # declare instance variables
    self.name = n

  # instance method requires self-reference
  def customSpeak(self, message):
    print(self.name + " says " + message)


  # static method requires no self-reference
  @staticmethod
  def speak(message):
    print(message)
```

```
class Person extends Object
{
  String name;      // declare instance variables

  Person()          // default constructor
  {  this.name = "Lee";  }

  Person(String n)  // parameterized constructor
  {  this.name = n;  }

  // instance method
  void customSpeak(String message)
  {
     System.out.println(this.name +
                        " says " + message);
  }

  // static method
  static void speak(String message)
  {  System.out.println(message);  }

}
```

```
# usage

p = Person("Percy")
p.name                      # returns "Percy"
p.customSpeak("hi")         # "Percy says hi"
Person.speak("goodbye")     # "goodbye"
```

```
// usage

Person p = new Person("Percy");
p.name                      // returns "Percy"
p.customSpeak("hi");        // "Percy says hi"
Person.speak("goodbye");    // "goodbye"
```

Miscellaneous

```
import time
time.sleep(3) // pause for 3 seconds
```

```
try { Thread.sleep(3000); } // pause 3000 millisec
catch (Exception ex) {    } // if interrupted...
```