|  |  |
| --- | --- |
| **Python** | **Java** |
| import math | import java.lang.Math; |
| # comment | // comment |
| x = 4 | int x = 4; |
| x = 5.0 | double x = 5.0; |
| x+=1 | x++; |
| 4 \* 5 | 4 \* 5 |
| 4 / 5 | 4.0 / 5.0 |
| 4 // 5 | 4 / 5 |
| math.sin(1) | Math.sin(1) |
| math.pi | Math.PI |
| x < 5 and is\_pink | x < 5 && isPink |
| not is\_pink | !isPink |
| s = None | s = null; |
| "s" is str | "s" == str |
| "s" == str | "s".equals(str) |

|  |  |
| --- | --- |
| **Python** | **Java** |
| scores = list()  – OR –  scores = [] | ArrayList scores = new  ArrayList<Integer>(); |
| scores.append(1) | scores.add(1) |
| my\_list = [1, 'hi', 5.0] |  |
| my\_tuple = (1,'hi',5.0)  # Tuples are immutable  # Can't edit | // No tuples in Java, but  // available in Guava…  // "ImmutableList" |
| a = [1,2,3,4,5] | long[] a = {1,2,3,4,5}; |
| x = a[0] # x is an int | long x = a[0]; |
| y = a[2:4] # y is a list | # not possible for an array |
| y = scores[1:3] | scores = scores.subList(1,3); |

# Lists and Tuples

# Arithmetic and Logic

|  |  |
| --- | --- |
| **Python** | **Java** |
| name = 'peter'  – OR –  name = "peter" | String name = "peter"; |
| # strings are immutable | // strings are immutable |
| s = 'a'  # no independent  # character type in Python | char c = 'a'  # can only store one char  # in a char |
| s = "abcde" | String s = "abcde"; |
| c = s[0] # c is a str! | char c = s.charAt(0); |
| b = s[2:4] # b is a str | String b = s.substring(2,4); |
| if isinstance(c,str): | if(b instanceof String) { |
| if type(c) is str: | if(b.getClass() ==  String.class) |
| length = len(s) | int len = s.length(); |
| d = s[-1] | char d = s.charAt(len-1) |
| b = s[-2:-1] | b = s.substring(len-2,len-1) |
| b = s[-2:] | b = s.substring(len-2,len) |
| b = s[:2]  – OR –  b = s[0:2] | b = s.substring(0,2) |

# Strings

|  |  |
| --- | --- |
| **Python** | **Java** |
| if a < b:  comp = 'less than'  elif a == b:  comp = 'equal to'  else:  comp = 'greater than'  print("a is",comp,"b") | if (a < b) {  comp = "less than";  } else if (a == b) {  comp = "equal to";  } else {  comp = "greater than";  }  *S.o.pl*("a is "+comp+" b") |
| x = 0  while x < 10:  x += 1 | long x = 0;  while (x < 10) {  x++  } |
| for x in range(5,10):  print(x) | for(int x = 5; x<10; x++) {  System.out.println(x)  } |
| y=f(x)  while not is\_good(y):  y=f(x) | do {  y=f(x)  } while(!isGood(y)) |

# Control Structures

# Encoding

# Control Structures

|  |  |
| --- | --- |
| **Python** | **Java** |
| ***Specify bits in a byte*** | ***Specify bits in a byte*** |
| b = b'\xff' | byte b = 0xff; |
| ***Specify bits in an int*** | ***Specify bits in an int*** |
| i = 0x7fffffff | int i = 0x7fffffff; |
| ***Parsing and formatting*** | ***Parsing and formatting*** |
| s = '1' | String s = "1"; |
| i = int(s) | int i = Integer.parseInt(s); |
| s = str(i) | s = ""+i; |
| ***Get bits used to store an int*** | ***Get bits used to store an int*** |
| b = i.to\_bytes(2,'big') | //Use ByteBuffer or streams |
| ***Load bits into an int*** | ***Load bits into an int*** |
| i2=int.from\_bytes(b,'big') | //Use ByteBuffer? |
| ***Encoding/Decoding Unicode*** | ***Encoding/Decoding Unicode*** |
| s2 = s[0] | char c = s.charAt(0); |
| i = ord(s2) | long i = c; |
| s3 = chr(i) | char c3 = i; |

|  |  |
| --- | --- |
| **Python** | **Java** |
| def greet(name):  return "Hello, " + name | public String greet(String name) {  return "Hello, " + name  } |
| def greet(name):  """  :param name: The name …  :return: A greeting …  """  return "Hello, " + name | /\*\*  @param name The name …  @return A greeting …  \*/  public String greet(String name) {  return "Hello, " + name  } |
| try:  raise Exception('Ow!')  except ZeroDivisionError:  print('Division by zero')  finally:  print('Cleanup') | try {  throw new Exception("Ow!")  } catch (ArithmeticException e) {  *S.o.pl*("Division by zero")  } finally {  *S.o.pl*("Cleanup")  } |

|  |
| --- |
| ***Get bits used to store an int*** |
| b = i.to\_bytes(length=2, byteorder='big') |
| ***Load bits into an int*** |
| i2 = int.from\_bytes(bytes=b, byteorder='big') |

# Python's Keyword Arguments