1 I/O

Java:

```java
System.out.println("Hello world");
System.out.println("Earth is #" + 1);
int num = sc.nextInt(); // Assuming that you've
    done the appropriate overhead
```

Python:

```python
print("Hello World")
print("Earth is #{}".format(1)) # Similar to printf
num = int(input('Enter a number: ')) # There is no
    overhead
```

2 if, else, and elif

Java:

```java
if(CONDITIONAL) {
    Stuff...
}
else if(CONDITIONAL) {
    Stuff...
}
else {
    Stuff...
}
```

Python:

```python
if CONDITIONAL:
    Stuff...
elif CONDITIONAL:
    Stuff...
else:
    Stuff...
```

3 Loops

3.1 while

Java:

```java
while(CONDITIONAL) {
    Stuff on repeat...
}
```

Python:

```python
while CONDITIONAL:
    Stuff on repeat...
```

3.2 C-Style for

Java:

```java
for(int i = 0; i < n; ++i) {
    Stuff on repeat n times...
}

for(int i = 0; i < a.length; ++i) {
    Use a[i]...
}
```

Python:

```python
for i in range(n):
    Stuff on repeat n times...

for i in range(len(a)):
    Use a[i]...
```

3.3 Python Style

```python
for ai in a:
    Stuff using ai ≡ a[i]...

OR

for i, ai in enumerate(a):
    Stuff using ai ≡ a[i]...
```

4 Booleans

Java:

```java
true
false
```

Python:

```python
True
False
```

Java:

Python:

```java
if(x > 1) {...
if(x >= 1) {...
if(x == 1) {...
if(x != 1) {...
if(!b) {...
if(b1 && (b2 || !b3)) {...
if(1 < x && x < 10) {...
if(x == y && y == z) {...
```

```python
if x > 1:...
if x >= 1:...
if x == 1:...
if x != 1:...
if not b:...
if b1 and (b2 or not b3):...
if 1 < x < 10:...
if x == y == z:...
```

5 Arithmetic

Java:

```java
x = 5;
x = x + 1; // x == 6
x += 1; // x == 6
x++; // or ++x; x == 6
x /= 2; // x == 2
N/A // Java does not support
N/A // Java does not support
N/A // Java does not support
x %= 2; // x == 1
```

Python:

```python
x = 5
x = x + 1 # x == 6
x += 1 # x == 6
N/A # Python does not support
x /= 2 # x == 2.5
x //= 2 # x == 2 (Integer division)
x = 5 ** 2 # 25 (5 * 5)
x **= 2 # x == 25 (x * x)
x %= 2 # x == 1
```

6 Methods / Functions

Java:

```java
ACCESS_MODIFIER RETURN_TYPE
    FUNCTION_NAME(PARAMETERS) {...

public int add(int x, int y) {
    return x + y;
}
```

Python:

```python
def FUNCTION_NAME(PARAMETERS):...

def add(x, y):
    return x + y
```

6.1 Overloading / Keyword Arguments

Java:

```java
public double calc(double x) {
    return calc(x, 5, 5);
}

public double calc(double x, double y, double z) {
    return x + (y / z);
}

...

calc(3); // returns 4
calc(3, 4, 2); // returns 5
```

Python:

```python
def calc(x, y=5, z=5):
    return x + (y / z)

...

calc(3) # returns 4
calc(3, 4) # returns 3.8
calc(3, 4, 2) # returns 5
calc(3, z=4) # returns 4.25
calc(3, y=8, z=4) # returns 5
calc(3, z=4, y=8) # returns 5
```

7 Classes

7.1 Class Declaration

Java:

Python:

```java
ACCESS_MODIFIER class CLASS_NAME extends
    PARENT_CLASS implements INTERFACES {...

public class MyClass extends ParentClass
    implements TheirInterface {...

public class MyClass {... // Implicitely extends
    Object class
```

```python
class CLASS_NAME(PARENT_CLASS1, PARENT_CLASS2,
    etc):...

class MyClass(ParentClass):

class MyClass(object): # Explicitely extends
    object class
```

7.2 Constructor and Methods

Java:

```java
public MyClass(int xIn, double yIn, boolean zIn) {
    // Assuming that MyClass extends a class that
        has a constructor with int x
    // Assuming that MyClass declared instance
        variables:
    //     double y;
    //     boolean z;
    super(xIn);

    y = yIn; // ≡ this.y = yIn;
    z = zIn; // ≡ this.z = zIn;
}

public get2y() {
    return y * 2; // ≡ return this.y * 2;
}
```

Python:

```python
def __init__(self, x_in, y_in, z_in):
    # Assuming that MyClass extends a class that
        has a constructor with int x
    super().__init__(x_in)

    self.y = y_in # ≢ y = y_in;
    self.z = z_in # ≢ z = z_in;

def get2y(self):
    return self.y * 2 # ≢ return y * 2
```

7.3 Initialization and Method Calling

Java:

```java
MyClass mc = new MyClass(2, 3.5, true);
System.out.println(mc.get2y());
```

Python:

```python
mc = MyClass(2, 3.5, True)
print(mc.get2y()) # Note that while self is an
    parameter I don't actually have to pass it
```

8 Include / Import

Java:

```java
// Not needed if file is in the same directory

// Reference things in the package as thing
include path.to.folder.package;
```

Python:

```python
# Needed always

# Reference thing as path.to.folder.module.thing
import path.to.folder.module

# Reference things in module as md.thing
import path.to.folder.module as md

# Reference thing as thing
from path.to.folder.module import thing

# If it is in the same directory
import module
```

8.1 Examples

Java:

Python:

```java
include java.util.Scanner;

Scanner sc = new Scanner(System.in);
```

```python
# Don't worry about what this does, it's an
# example of how to import

# This is the only style of import that I will use
import scipy.sparse
matrix = scipy.sparse.csr_matrix(range(10))

import scipy.sparse as sp_sparse
matrix = sp_sparse.csr_matrix(range(10))

from scipy.sparse from csr_matrix
matrix = csr_matrix(range(10))
```

9 Example Factorial Program

9.1 Java

FactorialCalculartor.java

```java
public class FactorialCalculator {
    public int calc(int n) {
        int result = 1;
        while(n > 0) {
            result *= n;
            n--;
        }

        return result;
    }
}
```

Factorial.java

```java
public class Factorial {
    public static void main(String[] args) {
        FactorialCalculator factCalc = new FactorialCalculator();

        for(int i = 1; i <= 10; ++i) {
            int fact = factCalc.calc(i);

            if(fact > 100) {
                System.out.println("Large Answer: " + fact);
            }
            else {
                System.out.println("Small Answer: " + fact);
            }
        }
    }
}
```

9.2 Python

factorial_calculator.py

```python
class FactorialCalculator(object):
    def calc(self, n):
        result = 1
        while n > 0:
            result *= n
            n -= 1

        return result
```

factorial.py

```python
import factorial_calculator


factCalc = factorial_calculator.FactorialCalculator()

for i in range(1, 11):
    fact = factCalc.calc(i)

    if fact > 100:
        print("Large Answer: {}".format(fact))
    else:
        print("Small Answer: {}".format(fact))
```