Python Ecosystem Solutions

1.

```
py import math print(math.factorial(5))
```

1.

```
py from math import factorial print(factorial(5))
```

1.

```py # module1.py from math import factorial

def get_factorial(n): return factorial(n) ```

```
py # module2.py from module1 import get_factorial print(get_factorial(5))
```

## Variable Scope

1. `a` is a local variable in the scope of `my_function` because it is an argument name. `b` is also a local variable inside my_function, because it is assigned a value inside `my_function`. `c` and `d` are both global variables. It doesn't matter that `d` is created inside an if block, because the inside of an if block is not a new scope - everything inside the block is part of the same scope as the outside (in this case the global scope). Only function definitions (which start with def) and class definitions (which start with class) indicate the start of a new level of scope.
2. Both `a` and `b` will be created every time `my_function` is called and destroyed when `my_function` has finished executing. `c` is created when it is assigned the value `3`, and exists for the remainder of the program's execution. `d` is created inside the if block (when it is assigned the value which is returned from the function), and also exists for the remainder of the program's execution.
3. If blocks are executed conditionally. If `c` were not greater than `3` in this program, the if block would not be executed, and if that were to happen the variable `d` would never be created.
4. We may use the variable later in the code, assuming that it always exists, and have our program crash unexpectedly if it doesn't. It is good to ensure that a variable always defined by assigning it some default value at the start. It is much easier and cleaner to check if a variable has the default value than to check whether it exists at all.
```