**Declarations:**

Declarations introduce new elements that facilitate computation or elements the program can use viz. variables, structures, arrays, functions, etc.

**Declarations in Go:**  
The designers of Go wanted to take a more readably inclined approach with Go, to get rid of the complexities of other primitive languages like C, C++.  
The declarations in golang follow a left to right declaration structure.   
Unlike C’s right to left declaration structure which goes on the lines of:

|  |
| --- |
| Int x;  Int a[12]  Int \*p |

Go on the other hand tries to be more on the readable side like:

|  |
| --- |
| X: variable of int;  A: array[3] of int;  P pointer of int |

Before the declaration of variables we add the keyword *var* to denote the declaration of variables.  
The keyword *var* declares one or more variables  
Taking redundant keywords away along with the colon boils down to:

|  |
| --- |
| <data type> <variable name>  Eg:  Var X ,Y int=1,2  Var A [3]int  Var P \*int |

The first example reads “variables X and Y of type int that are 1 and 2 respectively”.

Constant declarations go in the form:

|  |
| --- |
| Const <constant name>=<value of constant>  Eg:  Const Pi=3.14 |

The declaration structure facilitates in improving the readability of the code, hence eliminating complexity to an extent.  
This right to left declarations makes it easy because we’ve been reading the left to right structure since kindergarten.

**Declarations in Swift:**

The declarations of Swift isn’t too different from that of Go, the only difference is that Swift declarations employ a colon. The declarations go as follows:

|  |
| --- |
| Var X:int |

Swift follows the same left to right principle but at the same time employs the colon to read “of type”, so the example in the box reads “variable X of type int”.  
This coherently supports readability and thus gives us an idea why Swift has caught on to be so successful, because Readability is a major factor in determining the success of a program.  
Multiple variables of the same type can be declared the same way as Go;

|  |
| --- |
| Var X, Y, Z :int |

The above declaration reads “variables x, y, and z of type z”.

In conclusion to these two modern Programming languages, they’re very similar to each other, apart from some existential touches that each of them employ for themselves, like the ‘:’ in swift.

**Declarations in Python:**  
So in contrast to the other two beating it out in the comparison, Python is drastically different comparatively to the similarities between Swift and Go.  
Now the thing that makes Python SO different is that Python is.. wait for it.. DYNAMIC.

What does that mean?  
It means that Python does not need any declaration for its variables. There is no declaration at all, rather there’s assignment. To put it into perspective this is what a Python assignment looks like, and for the sake of brevity We’ll call it Pythons declaration.

|  |
| --- |
| A=100  B=100.2  X=”Hello” |

Now these assignments mean that the value of the left hand symbol is determined by the evaluation of the expressions on the right hand side. In this case A is evaluated to be 100, B is evaluated to be 100.2 which is subsequently stored as a floating point value, and X is kept as the name of the string “Hello”.  
The “Dynamic” thing about python is that the type of the variable is decided upon the assignment statement irrespective of its type in an instance before that particular assignment.   
Now to justify the use of Dynamic variable declaration OR assignment…  
What’d I’d like to think is that it makes the declaration process very fast and easy. It reduces the number of characters on the screen. No specifying the data types, the language is smart enough to figure out what the data type of a variable is supposed to be.  
But then the problem arises, that as the code becomes larger the previous data type of a variable gets lost as the same variable symbol is being assigned a new value.

**Functions**

**Functions in Go:**

The functions in Go are declared as follows

|  |
| --- |
| Func <function name>(<data types separated by commas>) <return type> |

The function declaration uses the keyword to initiate the function declaration process, followed by the function name, and the arguments passed inside parens after the function.  
Now, what makes Go different is where the return types are specified, that is right after the function declaration, i.e., after the argument specification.  
To make it simple, the function declaration of Go follows the same left to right principle of the variable declaration.

**Functions in Swift:**

The functions in Swift are defined as follows

|  |
| --- |
| Func <function name> (<variable name>:<data type>) -> <return type> {/...../} |

The declaration begins with the keyword func, which denotes the beginning of the declaration.  
Unlike Go specifying the name of the input parameter is necessary which is then followed by a colon and then the data type. And just like Go the return type is after the function signature, but, preceded by a return type arrow, which is a hyphen followed by a right angular bracket.

This like Go facilitates in readability at the same time utilizing extra elements to add to the readability like the colon that means ‘of type’, and the return type arrow.

**Functions in Python:**  
The function declaration in Python is as follows

|  |
| --- |
| Def <function name>(arguments): ….. return<variable opt> |

The function declaration begins with the keyword ‘def’ which followed by the function name and then the parenthesis which has the formal arguments and then the colon which denotes the beginning of a function. The body is then written under the signature with the proper indents denoting the scope block of the function. Then finally the return keyword returns the flow of the program to the function call, it also denotes the end of the function. Now the function can return a value to the function call if specified after return.

This uses the primitive way to declare a function apart from the ‘def’ keyword, which is obviously new and maybe even unique to Python.  
Okay now to address the elephant in that declaration method. Where is the return type?  
If you recall from the **Declarations in Python** section, we learn that the we have dynamic Variables that is the data type does not matter until the determination of the value on the right hand side of the ‘=’ sign.  
This makes Python dynamic, and in a way easy to use, and subsequently convenient. Convenience in the form of lesser lines of code and crucial time saving.