ENGF0002 (Design and Professional Skills)
Scenarios

The focus of this document is on the differences in function application between the three prototypes of the programming language. It follows an observation-explanation-conclusion structure in that an observation is laid down, explained, and a theory is formulated it out of it.

Classifier-1 (1.rkt)

Observation-1: When more parameters than required are passed on to a function, then Core-1 and Core-2 report errors. However, Core-3 compiles.

Theory-1: Assuming a function- $f(k_1, k_2, .. , k_n)$- that accepts n parameters. When i parameters, such that $i > n$, are passed to such a function, then Core-1 and Core-2 report errors and do not compile.

Example-
fun $f(k_1, k_2, .. , k_n)$:
  //function body
end
$f(k_1, k_2, .. , k_n , .. , k_i)$

Core-3, on the other hand, does not report *any* errors and compiles; taking only the required number of parameters.

Partition after test: {Core 1, Core 2}, {Core 3}

Classifier-2 (2.rkt)

Observation-2: When a variable- that is used in a function- is declared after it, then Core-1 and Core-3 report an error and do not compile. However, Core-2 compiles.

Theory-2: This means that Core-1 and Core-3 compile line-by-line and do not check whether a variable, that is used before initialisation, is declared later on.
My theory is confirmed after a test where a variable *x* (as shown in the classifier) is declared before *try()* compiles in Core-1 and Core-3

Example-

```
x = 2
fun try():
        print(x)
end
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Core-1: 2
Core-3: 2
```

Core-2, on the other hand, checks whether the variable is declared elsewhere and then performs the function on it.

Partition after test: {Core 1, Core 3} , {Core 2}

This test also raises an important similarity between the three prototypes: all three languages take the scope of any variable declared in the source code as global, except those that are declared in a function. This indicates that any variable in the source code can be used by any function without any access restriction.