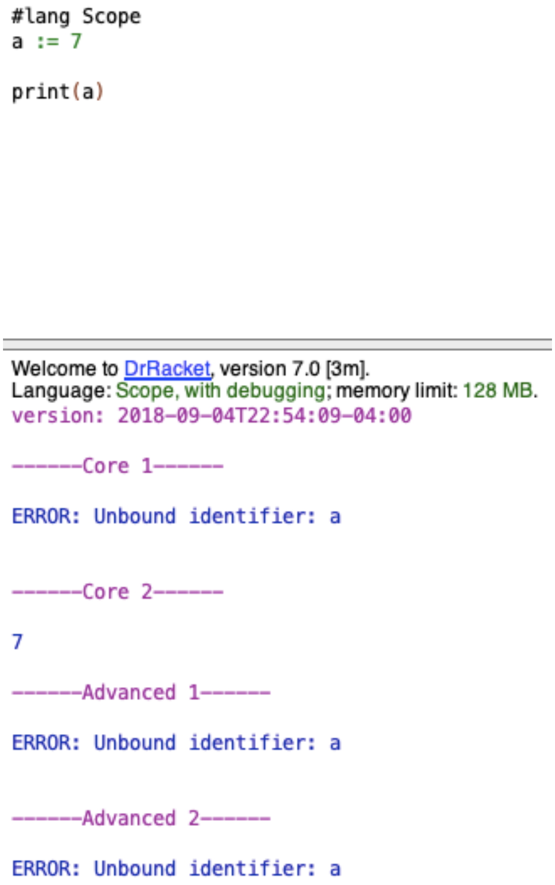
ENGF0002 (Design and Professional Skills)

Scenarios

The focus of this document is on the differences in variable assignment and scope between the four 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)



```
#lang Scope
a := 7

print(a)
```

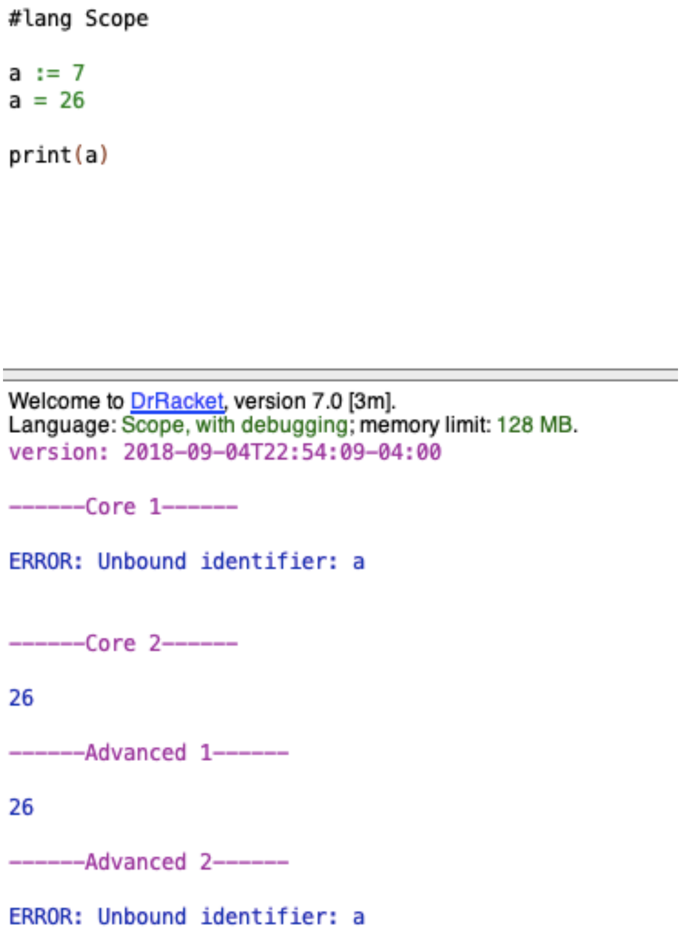
```
Welcome to DrRacket, version 7.0 [3m].
Language: Scope, with debugging; memory limit: 128 MB.
version: 2018-09-04T22:54:09-04:00

------Core 1------

ERROR: Unbound identifier: a


------Core 2------

7

------Advanced 1------

ERROR: Unbound identifier: a


------Advanced 2------

ERROR: Unbound identifier: a
```

Observation-1: When the assignment operator- := - is used before a variable is declared in the source code, then Core-1, Advanced-1 and Advanced-2 all report errors, while Core-2 compiles.

Theory-1: This means that Core-2 does not check whether a variable is declared before in the source code and simple uses the operator - := - to bind the new value to the variable. However, the other three prototypes do check whether *a* is declared before its value is changed, therefore, reporting an error in this case.

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

Classifier-2 (2.rkt)



```
#lang Scope

a := 7
a = 26

print(a)
```

```
Welcome to DrRacket, version 7.0 [3m].
Language: Scope, with debugging; memory limit: 128 MB.
version: 2018-09-04T22:54:09-04:00

-------Core 1-------

ERROR: Unbound identifier: a


-------Core 2-------

26

-------Advanced 1-------

26

-------Advanced 2-------

ERROR: Unbound identifier: a
```

Observation-2: Continuing from the previous observation, when the = operator is used after the := operator on the same variable name, then Core-2 and Advanced-1 compile, while Core-1 and Advanced-2, again, report errors.

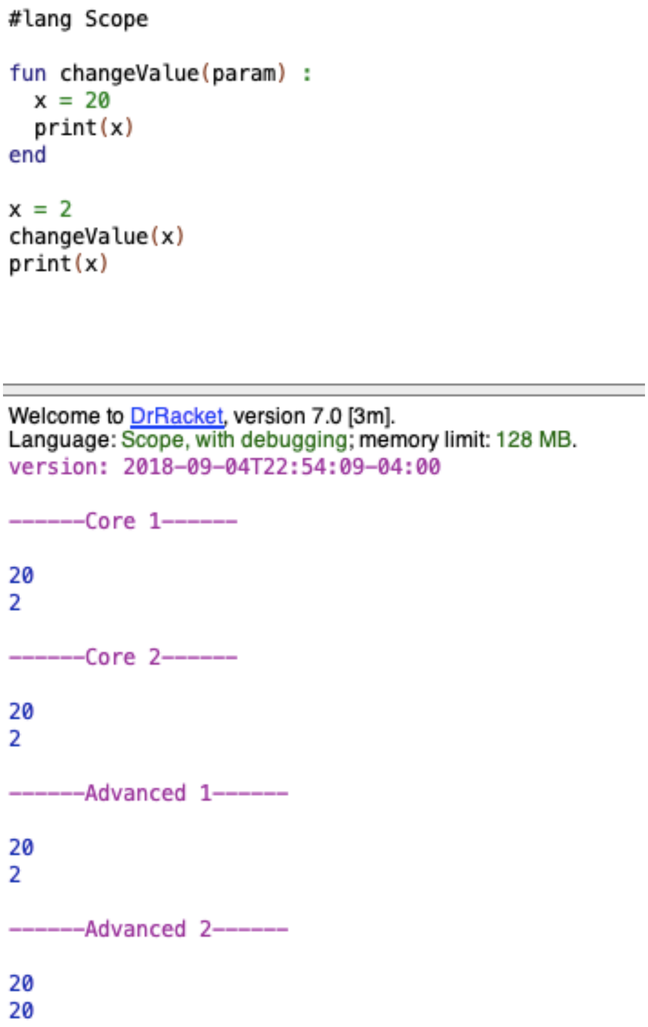
Theory-2: This implies that Core-2 uses the := operator and the = operator interchangeably and therefore, simply changes the value bound to the variable when the = operator is used after the := operator.

Advanced-2, however, does not use := to declare a variable- as shown in the previous classifier when it reported an error- instead, when *a* is printed, it simply searches for the value of *a* in the source code (which is declared using the = operator) and then prints it out.

Core-1 and Advanced-1, however, notice that the := operator is used before the variable is declared and hence, report an error and do not compile.

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

Classifier-3 (3.rkt)



```
#lang Scope

fun changeValue(param) :
  x = 20
  print(x)
end

x = 2
changeValue(x)
print(x)
```

```
Welcome to DrRacket, version 7.0 [3m].
Language: Scope, with debugging; memory limit: 128 MB.
version: 2018-09-04T22:54:09-04:00

------Core 1------

20
2

------Core 2------

20
2

------Advanced 1------

20
2

------Advanced 2------

20
20
```

Observation-3: When the value of a variable that is declared outside a function is changed inside a function, then Core-1, Core-2 and Advanced-1 do not change the original value of the variable, but Advanced-2 does.

Theory-3: When the value of *x* is changed inside the function, then Core-1, Core-2 and Advanced-1 do not keep the changes. This indicates that although the scope of the variable is global, and it can be manipulated in a function, there is a restriction on changing its value, which can only be done outside of the function (or where it is declared

For Advanced-2, on the other hand, this restriction is not imposed and hence, the value of the variable is permanently changed.

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

In first test: Core 2 differentiated from Core 1
In second test: Advanced 1 differentiated from Core 1
In third test: Advanced 2 differentiated from Core 1
Hence, all languages have been differentiated from each other.