

Manual de Usuario

Naomi Macias Honti A01282098

Vintage

23 nov. 20

## Contents

Estructura .....	3
Declaración de variables .....	3
Declaración de funciones .....	4
Estatutos.....	5
Asignación .....	5
Llamada .....	5
Lectura.....	5
Escritura .....	6
Decisión .....	6
Repetición .....	6
WHILE DO .....	7
DO WHILE .....	7
FROM TO .....	7
Función especial.....	7
Ejemplo de uso de las funciones especiales: .....	8
Resultado en el output gráfico: .....	9
Retorno.....	9
Evaluaciones .....	9
Expresiones .....	11
Ejecutar código Vintage .....	12
Programas .....	12
Visual Studio Code.....	12
Python .....	12
Archivo Vintage .....	12
Agrega tu código Vintage .....	13

## Estructura

El lenguaje sigue la siguiente estructura:

```
Program <variable>;  
<declaracion de variables globales>  
  
<declaracion de funcion>  
  
main()  
{  
    <estatutos>  
}
```

Primero tenemos la declaración del nombre del programa, luego una declaración de variables globales, seguido de una declaración de funciones y finalmente un main.

<variable>

Las variables pueden estar conformadas por números y letras, pero obligatoriamente tiene que empezar con una letra.

## Declaración de variables

Las variables locales y globales siguen la misma estructura:

```
var  
    <tipo> <variable>, <variable>, ... <variable>;  
    <tipo> <variable>;  
    ...
```

<tipo>

Vintage permite declarar variables del tipo int, float y char. Ejemplo:

```
var  
    int numerico, otroNumerico;  
    float decimal;  
    char caracter;
```

NOTA: la declaración de variables, tanto globales como locales, son opcionales, pero solo se pueden utilizar variables declaradas.

NOTA: las variables se declaran con un nombre único que no puede ser repetido, el único caso es que se puede dar es que dos o más funciones tengan variables locales repetidas. Esto solo se permite porque son variables locales que no se mezclan.

## Declaración de funciones

Las funciones siguen la siguiente estructura:

```
module <tipoFuncion> <variable>(<parametros>);  
<declaracion de variables locales>  
{  
    <estatutos>  
}
```

<tipoFuncion>

Vintage permite declarar funciones del tipo: void, int, char y float. Ejemplo:

```
module void fun1();  
{  
    <estatutos>  
}
```

<parametros>

Los parámetros en las declaraciones de las funciones tienen la siguiente estructura:

```
(<tipo> <variable>, <tipo> <variable>, ... <tipo> <variable>)
```

NOTA: Los parámetros se consideran variables locales de la función.

NOTA: La declaración de variables locales tiene la misma estructura que las variables globales.

Las funciones que no sean void, deben de tener una instrucción de retorno, Vintage no marcará error si no se cuenta con una instrucción de retorno, pero si, si la función no regresa un valor.

Ejemplo de una función con retorno:

```
module float fun2(int parametro);  
var  
    float temporal;  
{  
    <estatutos>  
    <retorno>  
}
```

## Estatutos

Vintage maneja diferentes tipos de estatutos clasificados en: asignación, llamada, lectura, escritura, decisión, repetición, función especial y retorno.

Los estatutos tienen la siguiente estructura:

```
<estatuto>  
<estatuto>  
<estatuto>  
...
```

### Asignación

Para asignar un valor a una variable se tiene que usar el estatuto Asignación.

Una asignación tiene la siguiente estructura:

```
<variable> = <expresion>;
```

La variable tiene que haber sido declarada y la expresión debe de dar como resultado un valor compatible con el tipo de variable.

Si la variable es del tipo char, solo aceptara otro char.

Si la variable es del tipo int, solo aceptara otro int.

Si la variable es del tipo float, solo aceptara otro float o un int.

### Llamada

Vintage maneja dos tipos de llamadas a funciones: con retorno y sin retorno. Las llamadas con retorno se manejan dentro de las expresiones. Las llamadas sin retorno tienen la siguiente estructura:

```
<variable>(<parametrosLlamada>);
```

NOTA: solo las funciones de tipo void pueden ser llamadas en una llamada sin retorno.

<parametrosLlamada>

Los parámetros en las llamadas a funciones tienen la siguiente estructura:

```
(<expresion>, <expresion>, ... <expresion>)
```

### Lectura

Para recibir un dato del usuario se tiene que utilizar el estatuto Lectura.

La lectura tiene la siguiente estructura:

```
read(<variable>, <variable>, ... <variable>);
```

NOTA: Vintage solo permite recibir un dato del usuario del mismo tipo que la variable donde se espera guardar el dato.

## Escritura

Para imprimir un valor en la consola se tiene que utilizar el estatuto Escritura.

La escritura tiene la siguiente estructura:

```
write(<parametroWrite>, <parametroWrite>, ... <parametroWrite>);
```

<parametroWrite>

Se pueden enviar dos tipos de datos por Escritura: <string> o <expresion>.

<string>

Inicia y termina con un " y contiene de 0 a más caracteres. Ejemplo:

```
write("Hello World!");
```

## Decisión

Los estatutos de decisión, también conocidos como IFs, tienen la siguiente estructura:

```
if(<evaluacion>) then {  
    <estatutos>  
}  
<else>
```

<else>

Tiene la siguiente estructura:

```
else {  
    <estatutos>  
}
```

NOTA: El uso de <else> es opcional, se puede eliminar.

## Repetición

Vintage maneja tres tipos de estatutos de repetición: dos son condicionales y uno es nocondicional. Son mejor conocidos como los LOOPS WHILE DO, DO WHILE y FROM TO

## WHILE DO

```
while(<evaluacion>) do {  
    <estatutos>  
}
```

## DO WHILE

```
do {  
    <estatutos>  
} while(<evaluacion>)
```

## FROM TO

```
from <variable> = <ecuacion> to <ecuacion> do {  
    <estatutos>  
}
```

## Función especial

Vintage cuenta con un output grafico que le permite al usuario dibujar figuras. Todas las funciones que manejan el output grafico son conocidas como funciones especiales. En el output grafico se muestra una tortuga que el usuario puede comandar avanzar o girar, conforme la tortuga avanza deja tras ella una línea, el usuario puede especificar de antemano el color y el grosor de la línea. El usuario también cuenta con una función que le permite dejar un punto en la posición actual de la tortuga.

Listado de funciones especiales:

PenUp()

Deja de pintar la línea aun si la tortuga avanza.

PenDown()

Comienza a pintar la línea desde la posición actual de la tortuga hasta donde avance. Es la forma default en la que comienza el output gráfico.

Point()

Deja dibujado un punto en la posición actual de la tortuga.

Color(<string>)

Define el color de la tortuga y por consiguiente el color de la línea. Los colores se definen en un string, puede ser "yellow", "green" o cualquier otro color en inglés.

Line(<int>)

Hace avanzar a la tortuga hacia la dirección en que este mirando. La longitud de su avance depende del valor enviado. La medición es en pixeles.

Turn(<int>)

Hace girar a la tortuga hacia la izquierda. Los grados de su giro dependen del valor enviado.

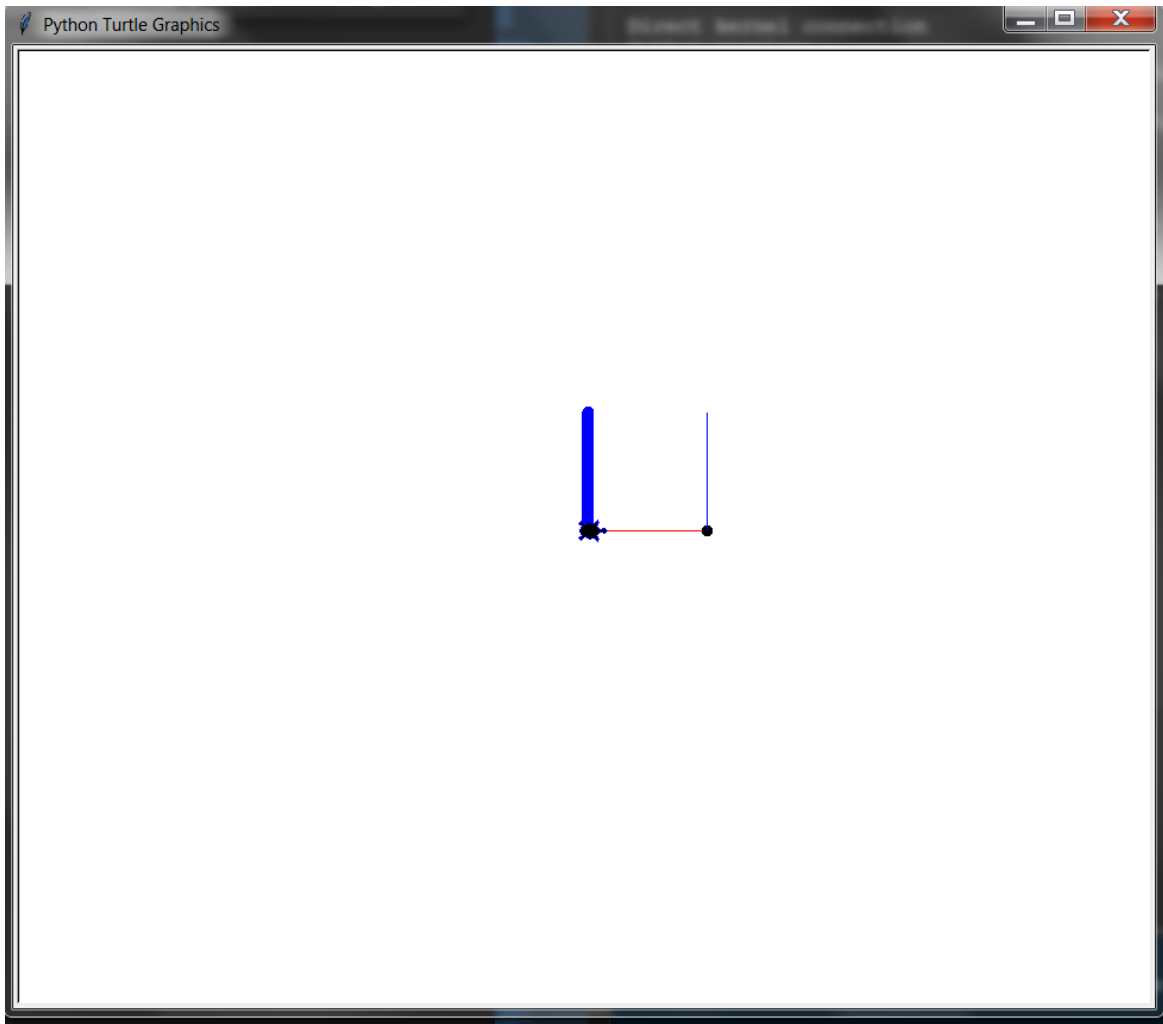
Ejemplo de uso de las funciones especiales:

```
Program MeMySelf;

main()
{
    Color("red");
    Line(100);
    Turn(90);
    Color("blue");
    Point();
    Line(100);
    Turn(90);
    PenUp();
    Line(100);
    Turn(90);
    PenDown();
    Size(10);
    Line(100);
    Turn(90);
}
```



Resultado en el output gráfico:



## Retorno

Tiene la siguiente estructura:

```
return(<expresion>);
```

NOTA: Vintage marcará un error si existe un estatuto de return dentro de una función void.

## Evaluaciones

<evaluacion>

Tienen la siguiente estructura:

```
(<comparacion> <agregarComparacion>)
```

<agregarComparacion>

Para agregar otra comparación a la evaluación se tiene que agregar con esta estructura:

```
(<comparacion> <signoEvaluacion> <comparacion>)
```

NOTA: El uso de <agregarComparacion> es opcional, se puede eliminar.

<signoEvaluacion>

Se permiten dos signos para evaluar: <and> y <or>.

<and>

Utiliza el símbolo &. Ejemplo de uso:

```
if(<comparacion> & <comparacion>) then {
```

<or>

Utiliza el símbolo |. Ejemplo de uso:

```
if(<comparacion> | <comparacion>) then {
```

<comparación>

Sirve para comparar dos valores entre sí, sigue la siguiente estructura:

```
(<expresion> <signoComparacion> <expresion>)
```

<signoComparacion>

Vintage maneja los siguientes signos de comparación: <igual>, <diferente>, <mayorIgual>, <mayor>, <menorIgual> y <menor>.

<igual>

Utiliza el símbolo ==. Ejemplo de uso:

```
if(<expresion> == <expresion>) then {
```

<diferente>

Utiliza el símbolo !=. Ejemplo de uso:

```
if(<expresion> != <expresion>) then {
```

<mayorIgual>

Utiliza el símbolo >=. Ejemplo de uso:

```
if(<expresion> >= <expresion>) then {
```

<mayor>

Utiliza el símbolo >. Ejemplo de uso:

```
if(<expresion> > <expresion>) then {
```

<menorIgual>

Utiliza el símbolo <=. Ejemplo de uso:

```
if(<expresion> <= <expresion>) then {
```

<menor>

Utiliza el símbolo <. Ejemplo de uso:

```
if(<expresion> < <expresion>) then {
```

## Expresiones

Expresiones se divide en dos opciones: <char> o <ecuacion>.

<char>

Inicia y termina con un ' y contiene solo un carácter. Ejemplo:

```
'c'
```

<ecuacion>

Tiene la siguiente estructura:

```
<termino> <agregarTermino>
```

<termino>

Un término puede ser una constante, de tipo int o float, una variable, una función o una ecuación entre paréntesis.

Ejemplo de términos:

```
2  
2.5
```

```
variable  
funcion()  
(<ecuacion>)
```

<agregarTermino>

Tiene la siguiente estructura:

```
<signo> <termino> <agregarTermino>
```

NOTA: El uso de <agregarTermino> es opcional, se puede eliminar.

<singo>

Los signos aceptados son +, -, \* y /.

Ejemplo de una ecuación:

```
(fun2() + 5 * var1) / 4
```

## Ejecutar código Vintage

### Programas

Para poder ejecutar Vintage necesitamos de tres cosas indispensables:

### Visual Studio Code

es un editor de código fuente desarrollado por Microsoft.

Link de descarga: <https://code.visualstudio.com/>

### Python

Es un lenguaje de programación que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

Link de descarga: <https://www.python.org/downloads/>

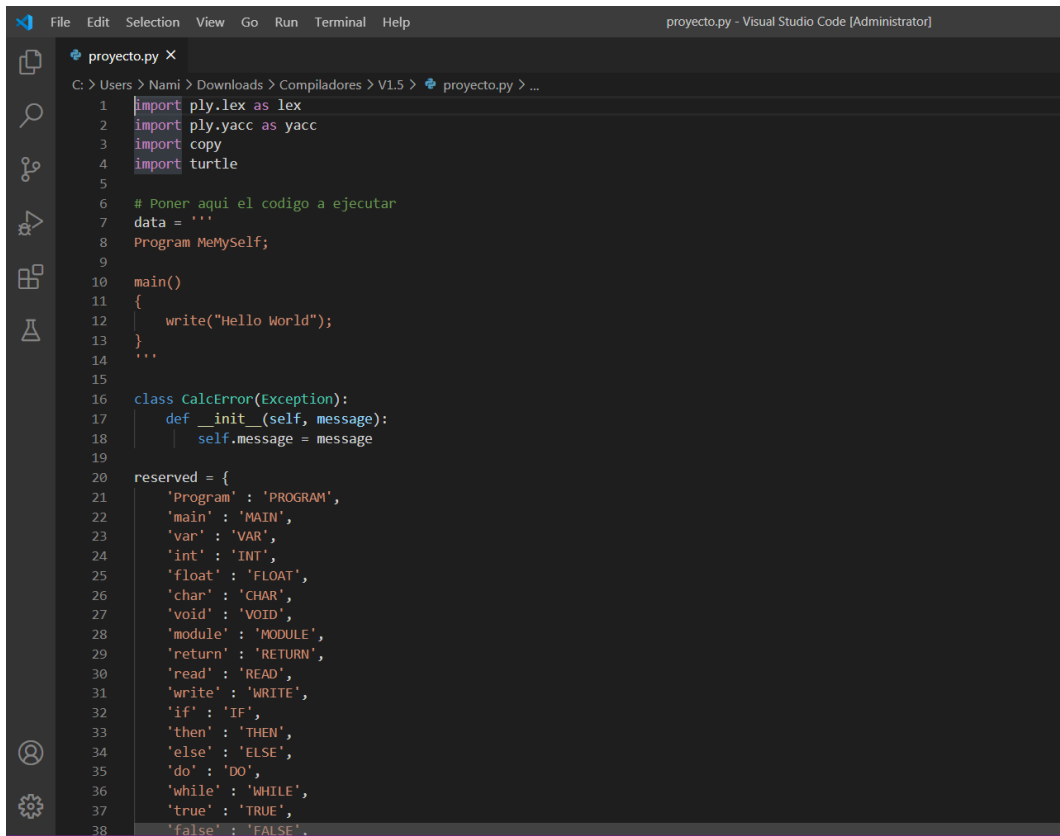
### Archivo Vintage

Contiene las instrucciones de código para ejecutar código Vintage.

Link de descarga: <https://github.com/NaomiMH/Clases.Semestre9.DC/blob/main/proyecto.py>

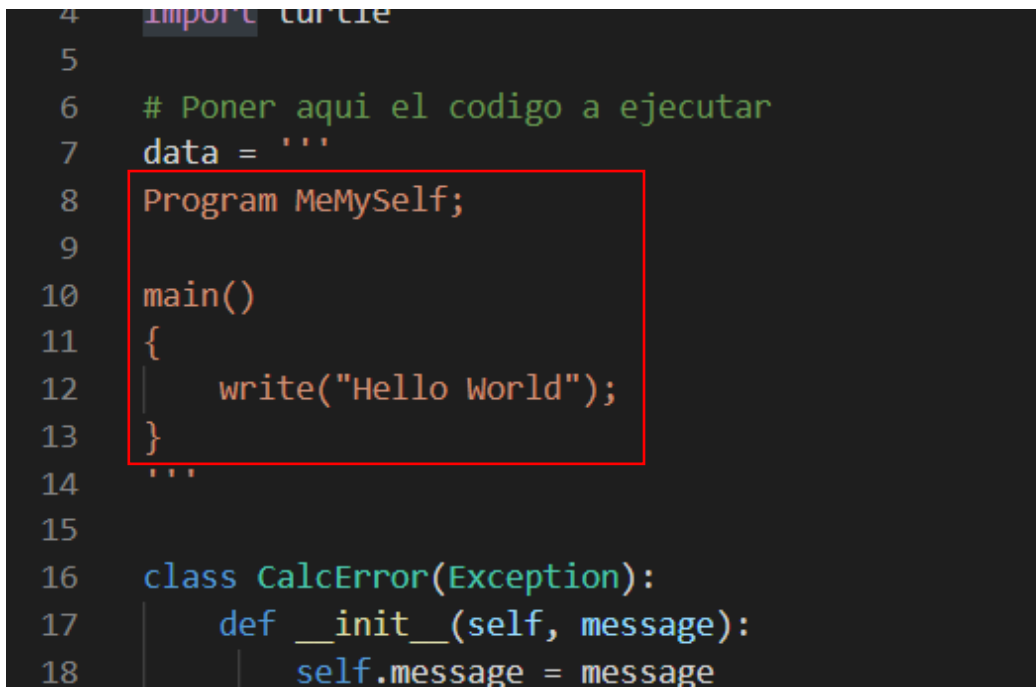
## Agrega tu código Vintage

Al abrir el archivo Vintage con Visual Studio nos aparecerá el siguiente programa:

A screenshot of the Visual Studio Code editor interface. The title bar at the top reads 'proyecto.py - Visual Studio Code [Administrator]'. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The left sidebar shows the Explorer, Search, Source Control, Run and Debug, and Extensions views. The main editor area displays the file 'proyecto.py' with the following Python code:

```
1 import ply.lex as lex
2 import ply.yacc as yacc
3 import copy
4 import turtle
5
6 # Poner aqui el codigo a ejecutar
7 data = ''
8 Program MeMySelf;
9
10 main()
11 {
12     write("Hello World");
13 }
14 '''
15
16 class CalcError(Exception):
17     def __init__(self, message):
18         self.message = message
19
20 reserved = {
21     'Program' : 'PROGRAM',
22     'main' : 'MAIN',
23     'var' : 'VAR',
24     'int' : 'INT',
25     'float' : 'FLOAT',
26     'char' : 'CHAR',
27     'void' : 'VOID',
28     'module' : 'MODULE',
29     'return' : 'RETURN',
30     'read' : 'READ',
31     'write' : 'WRITE',
32     'if' : 'IF',
33     'then' : 'THEN',
34     'else' : 'ELSE',
35     'do' : 'DO',
36     'while' : 'WHILE',
37     'true' : 'TRUE',
38     'false' : 'FALSE',
```

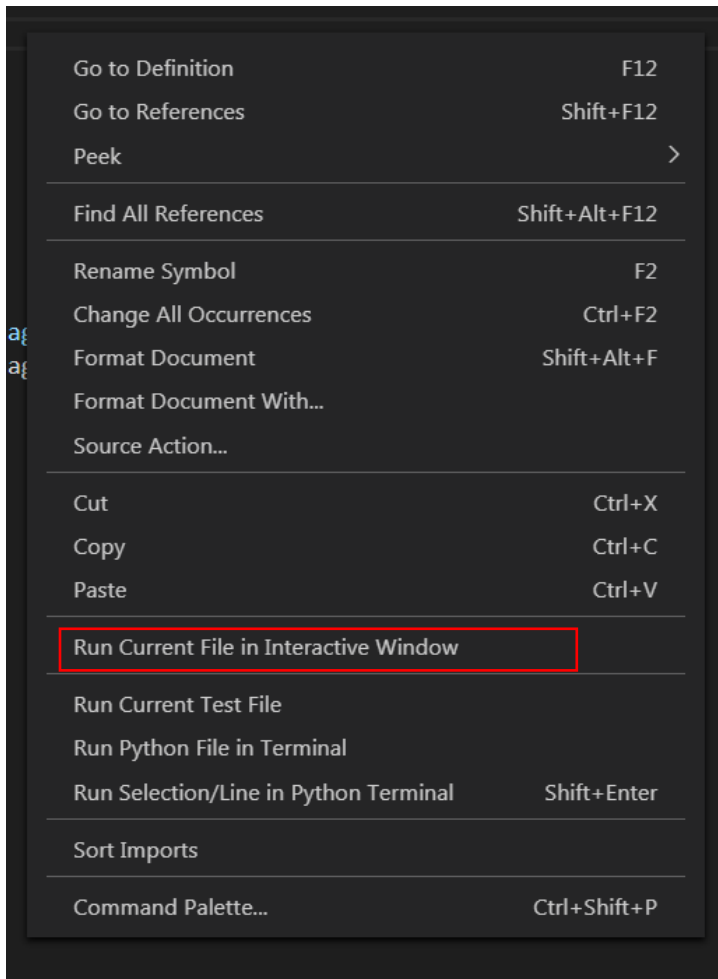
En la siguiente parte colocaremos nuestro código Vintage:

A close-up screenshot of the code in 'proyecto.py'. A red rectangular box highlights the code between lines 10 and 13, which is the main function. The code shown is:

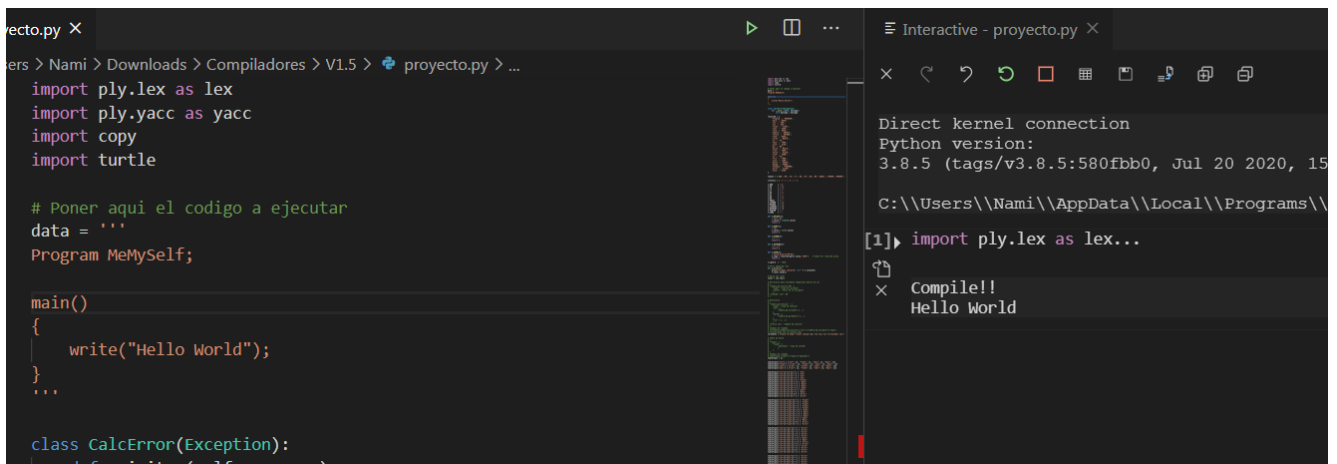
```
10 main()
11 {
12     write("Hello World");
13 }
```

Puede ser tan extenso como queramos, para este ejemplo dejaremos el Hello World.

A continuación, dar click derecho sobre el código y seleccionamos la opción “Run Current File in Interactive Window”.



Y listo. Se nos abrirá una ventana con el resultado de la ejecución:



NOTA: En caso de que la ejecución requiera el output gráfico, se recomienda cerrar la ventana Interactive antes de cada ejecución.

Cuando utilicemos el output gráfico, este se abrirá en una ventana extra:

