

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ingeniería en Ciencias y Sistemas  
Organización de Lenguajes y Compiladores 2  
Vacaciones Primer Semestre 2023

**Catedrático:** Ing. Luis Espino

**Tutor académico:** Diego Obín



**USAC**  
TRICENTENARIA  
Universidad de San Carlos de Guatemala

# PyTypeCraft

## Primer proyecto de laboratorio

<b>Competencias</b>	<b>3</b>
Competencia general	3
Competencias específicas	3
<b>Descripción</b>	<b>4</b>
Descripción General	4
Flujo específico de la aplicación	4
Ingreso de código fuente	5
Ejecución	5
Generación de reportes	5
<b>Componentes de la aplicación</b>	<b>5</b>
<b>Sintaxis de PyTypeCraft</b>	<b>7</b>
Generalidades	7
Tipos de dato válidos	8
Expresiones	9
Aritméticas	9
Multiplicación	10
División	11
Potencia	11
Módulo	12
Nativas	12
Relacionales	13
Lógicas	14
Impresión	15
Asignaciones	15
Funciones	17

Creación de funciones	17
Funciones Nativas	17
Llamada a funciones	17
Paso por valor o por referencia	18
Condicionales	18
Loops	19
Ciclo while	19
Ciclo for	19
Sentencias de transferencia	20
Arreglos	21
Structs	22
Tabla de Resumen de instrucciones	23
<b>Reportes generales</b>	<b>24</b>
Tabla de Símbolos	24
Árbol de análisis sintáctico	25
Tabla de errores	25
<b>Entregables y Calificación</b>	<b>26</b>
Entregables	26
Restricciones	27
Consideraciones	27
Calificación	28
Entrega de proyecto	28

**Miércoles 13 de junio hasta las 23:59 P.M.**

# 1. Competencias

## 1.1. Competencia general

Que los estudiantes apliquen los conocimientos adquiridos en el curso para la construcción de un intérprete utilizando las herramientas establecidas.

## 1.2. Competencias específicas

- Que los estudiantes utilicen herramientas para la generación de analizadores léxicos y sintácticos.
- Que los estudiantes apliquen los conocimientos adquiridos durante la carrera y el curso para el desarrollo de la solución.
- Que los estudiantes realicen análisis semántico e interpretación del lenguaje PyTypeCraft

## 2. Descripción

### 2.1. Descripción General

TypeScript es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, lo que significa que todo código JavaScript válido es también código TypeScript válido. Sin embargo, TypeScript añade características adicionales a JavaScript, incluyendo tipado estático y características de la programación orientada a objetos. Es por eso por lo que se le solicita el desarrollo de PyTypeCraft, un lenguaje de programación basado en Typescript que se podrá programar y ejecutar desde cualquier navegador.

### 2.2. Flujo específico de la aplicación

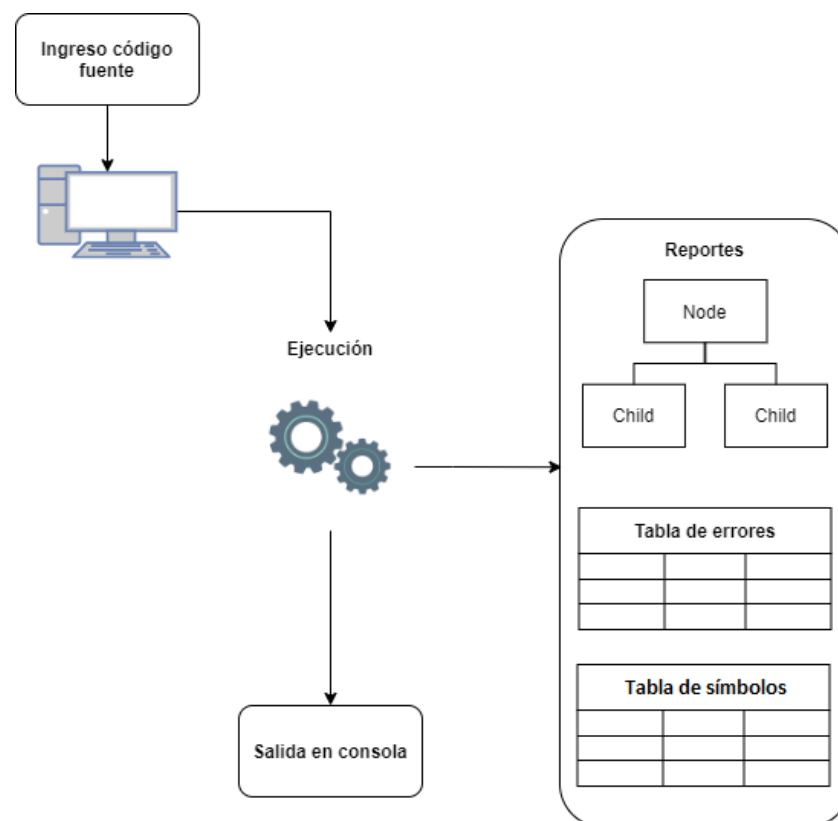


Ilustración 1: Flujo específico de la aplicación

## 2.3. Ingreso de código fuente

El lenguaje PyTypeCraft está basado en Typescript con instrucciones limitadas. Se deberá de contar con un lugar en la página donde los desarrolladores puedan programar y ejecutar sus archivos de PyTypeCraft para luego ejecutarlos.

## 2.4. Ejecución

Si el código se ejecuta y encuentra errores en el mismo entonces se deberá contar con recuperación de errores, los únicos errores de recuperación será tipo semántico, esto con la finalidad de que se pueda seguir ejecutando el resto del código de entrada y finalizando mostrar dichos errores en los reportes respectivos, de lo contrario se generará la salida en consola satisfactoriamente.

## 2.5. Generación de reportes

El código fuente se colocará en la página web, ya sea escribiendo desde la página o copiándolo y pegándole de un archivo .ts hacia la página. El programa se ejecutará y mostrará los resultados en la consola. Luego de eso, el desarrollador podrá ver distintos reportes que se generaron con el código que ingresó. Entre los cuales se encuentran el AST, Tabla de errores y la Tabla de símbolos. Estos reportes serán para que se verifique cómo el estudiante usa las estructuras internas para la interpretación del lenguaje. En la sección de reportes se detallará más al respecto de estos reportes.

# 3. Componentes de la aplicación

La aplicación deberá contar con las siguientes vistas, con la libertad de diseñar a su creatividad.

- **Página de bienvenida:** En esta vista deberá mostrar sus datos personales.
- **Página de análisis:** Contiene el editor de entrada y consola de salida. No hace falta abrir archivos, basta con copiar y pegar la entrada.
- **Página de Reportes:** En esta vista se podrán consultar los reportes. Los reportes de errores y tabla de símbolos se deben mostrar en la aplicación. El reporte de AST se debe descargar en un formato en el que se vea claramente el contenido, entre los formatos considerados para mostrar los reportes son png o svg.

Ilustración 2: Página de bienvenida

**Miércoles 13 de junio hasta las 23:59 P.M.**

PyTypeCraft Home Análisis Reportes

Input:

1	
2	
3	
4	

Output:

1	
2	
3	
4	

Analizar

Ilustración 3: Vista de análisis

PyTypeCraft Home Análisis Reportes

## Reportes del último análisis

A continuación se muestran los reportes generados por el más reciente análisis.

Reporte de errores Reporte de Tabla de símbolos AST

Ilustración 4: Página de consulta de reportes

## 4. Sintaxis de PyTypeCraft

PyTypeCraft provee distintas funcionalidades de Typescript, aun así, al tener funciones limitadas de este lenguaje se debe de definir bien la sintaxis de este al ser Typescript un lenguaje bastante amplio. También, para conocer más sobre la sintaxis de PyTypeCraft, puede revisar en la documentación de lenguaje de programación Typescript, pero con las limitaciones que en el proyecto se describe, visite los siguientes enlaces donde encontrará documentación oficial <https://www.typescriptlang.org/docs/>

**Miércoles 13 de junio hasta las 23:59 P.M.**

## 4.1. Generalidades

- **Comentarios.** Un comentario es un componente léxico del lenguaje que no es tomado en cuenta en el analizador sintáctico. Pueden ser de una línea (`//`) o de múltiples líneas (`/* ... */`).
- **Case Sensitive.** Esto quiere decir que distinguirá entre mayúsculas y minúsculas.
- **Identificadores.** Un identificador de PyTypeCraft debe comenzar por una letra `[A-Za-z]` o guión bajo `[_]` seguido de una secuencia de letras, dígitos o guión bajo.
- **Fin de instrucción.** Se hace uso del símbolo `“,”` para establecer el fin de una instrucción.

## 4.2. Tipos de dato válidos

PyTypeCraft únicamente aceptará los siguientes tipos de datos, cualquier otro no se deberá tomar en cuenta:

- **Null:** se representa con la palabra reservada *null*. Indica que no existe ningún valor.
- **Number:** valores numéricos enteros y decimales. Por ejemplo: 3,2,-1, 3.1416, 1.68...
- **Boolean:** valores booleanos, true o false.
- **String:** Cadenas de texto definidas con comillas dobles.
- **Any:** Cualquier tipo de dato.
- **Arreglos:** Conjunto de valores indexados entre 0 hasta n. Puede almacenar diferentes tipos. Para más información, consulte la sección 4.9.

```
[10, 20, 30, 40];  
[“Hola”, “Mundo”];  
[‘a’, 2.0, 5, [“Hola”, “Mundo”]];
```

- **Interface:** Estos son tipos compuestos definidos por el programador. Para mayor detalle, consulte la sección 4.10.

```
interface Rectangulo {  
    base: number;  
    altura: number;  
};
```

## 4.3. Expresiones

### 4.3.1. Aritméticas

Una operación aritmética está compuesta por un conjunto de reglas que permiten obtener resultados con base en expresiones que poseen datos específicos durante la ejecución. A continuación se definen las operaciones aritméticas soportadas por el lenguaje.

#### Suma

La operación suma se produce mediante la suma de número o strings concatenados.

Operandos	Tipo resultante	Ejemplos
Number + Number	Number	$2 + 3.3 = 5.3$ $2.3 + 8 = 10.3$ $1.2 + 5.4 = 6.6$
String + String <i>Nota: Number puede ser convertido a string con la función nativa "toString" para ser utilizados en esta operación.</i>	String	"hola"+"mundo"="holamundo" "Hola" + String(8) = "Hola8"
string.toUpperCase()	String	animal = "Tigre"; console.log(animal.toUpperCase()); #TIGRE
string.toLowerCase()	String	animal = "Tigre"; console.log(animal.toLowerCase()); #tigre



## Resta

La resta se produce cuando se sustraen el resultado de los operadores, produciendo su diferencia.

Operandos	Tipo resultante	Ejemplos
Number - Number	<b>Number</b>	$2 + 3.3 = -1.3$ $2.3 + 8 = -5.7$ $1.2 + 5.4 = -4.2$

## Multiplicación

El operador multiplicación produce el producto de la multiplicación de los operandos.

Operandos	Tipo resultante	Ejemplos
Number * Number	<b>Number</b>	$2 * 3.3 = 6.6$ $2.3 * 8 = 18.4$ $1.2 * 5.4 = 6.48$

## División

El operador división se produce el cociente de la operación donde el operando izquierdo es el dividendo y el operando derecho es el divisor.

Operandos	Tipo resultante	Ejemplos
Number / Number	<b>Number</b>	$2 / 3.3 = 0.60$ $2.3 / 8 = 0.2875$ $1.2 / 5.4 = 0.222$

**Miércoles 13 de junio hasta las 23:59 P.M.**

## Potencia

El operador de potenciación devuelve el resultado de elevar el primer operando al segundo operando de potencia.

Operandos	Tipo resultante	Ejemplos
Number ^ Number	<b>Number</b>	$2 ^ 3.5 = 11.31$ $2.3 ^ 8 = 783.10$ $1.2 ^ 5.4 = 2.67$

## Módulo

El operador módulo devuelve el resto que queda cuando un operando se divide por un segundo operando.

Operandos	Tipo resultante	Ejemplos
Number % Number	<b>Number</b>	$2 \% 3.5 = 2.0$ $2.3 \% 8 = 2.3$ $1.0 \% 5.0 = 1.0$

## Nativas

Typescript cuenta con una gran variedad de funciones nativas, sin embargo, PyTypeCraft únicamente contará con las siguientes funciones nativas:

Nombre	Símbolo o función	Descripción	Ejemplo
<b>Aproximación</b>	toFixed(number)	La función formatea un número usando notación de punto fijo.	let n: number = 10.156 n.toFixed(2) // 10.16

**Miércoles 13 de junio hasta las 23:59 P.M.**

<b>Pasar a Exponencial</b>	toExponential	Devuelve una cadena que representa el número en notación exponencial.	let n : number = 123.456 ; n.toExponential(2) ; // "1.23+2"
<b>Convertir a String</b>	toString	Devuelve una cadena que representa cualquier tipo convertido a string	let n : number = 123.456 ; n.toString(); // "123.456"
<b>Convertir a minúsculas</b>	toLowerCase	Convierte todas las letras en minúsculas	let n : string = "HOLA"; n.toLowerCase(); // "hola"
<b>Convertir a mayúsculas</b>	toUpperCase	Convierte todas las letras en mayúsculas	let n : string = "hola"; n.toUpperCase(); // "HOLA"
<b>Separador</b>	split	Divide una cadena en un array de cadenas	let n: string = "hola, mundo"; n.split(","); // [ "hola", "mundo" ]
<b>Concatenación</b>	concat	Combina 2 o más arrays	let arr: Array<number> = [1,2,3]; arr.concat([4,5,6]) // [1,2,3,4,5,6]

**Miércoles 13 de junio hasta las 23:59 P.M.**

### 4.3.2. Relacionales

Operador	Descripción
>	Mayor que: Compara ambos lados y verifica si el izquierdo es mayor que el derecho
<	Menor que: Compara ambos lados y verifica si el derecho es mayor que el izquierdo
>=	Mayor o igual que: Compara ambos lados y verifica si el izquierdo es mayor o igual que el derecho
<=	Menor o igual que: Compara ambos lados y verifica si el derecho es mayor o igual que el izquierdo
===	Igualación: Compara ambos valores y verifica si son iguales
!=	Distinto: Compara ambos lados y verifica si son distintos

#### EJEMPLOS:

Operandos	Tipo resultante	Ejemplos
Number[>, <, >=, <=] Number String [>, <, >=, <=] String	<b>Bool</b>	4 < 4.3 = true 4.3 > 4 = true 4.3 <= 4.3 = true 4 >= 4 = true "hola" > "hola" = false

### 4.3.3. Lógicas

Los siguientes operadores booleanos son soportados en PyTypeCraft. No se aceptan valores missing values ni operadores bitwise.

Operación lógica	Operador
OR	
AND	&&
NOT	!

**Miércoles 13 de junio hasta las 23:59 P.M.**

A	B	A && B	A    B	!A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	true	false	true

## 4.4. Impresión

Para mostrar información en la consola o en los reportes, PyTypeCraft cuenta con una forma para imprimir dependiendo de lo que deseamos realizar.

```
console.log("+", "-");           # Imprime + -
console.log("El resultado de 2 + 2 es $(2 + 2)"); # Imprime el resultado de 2 + 2 es 4
console.log("$a $(b[1])");      # Imprime el valor de a y el valor de b[1]
```

## 4.5. Asignaciones y Declaraciones

Una variable, en PyTypeCraft, es un nombre asociado a un valor. Una variable no puede cambiar su tipo.

La asignación se puede realizar de la siguiente forma:

```
let ID : TIPO = Expresión ;
ó
let ID = Expresión;
```

El sufijo **:TIPO** es opcional. Su función es asegurar que la expresión sea del tipo deseado. En caso la expresión sea distinta al tipo debe marcar un error.

```
let x : number = (3*5);      # Correcto
let str : number = "Saludo"; # ERROR: expected Number, got String
let var1 : string = true;   # ERROR: expected String, got Bool
let var = 1234;             # Correcto (aquí se asigna implícitamente el tipo number)
```

Para el manejo de variables “locales” y “globales” se deberá de distinguir el límite del bloque en el que se encuentra declarada y así determinar si la variable es global o solamente local

```
1 // Ejemplo 1: Entornos. Variables globales y locales.
2 let x:number = (3*5);// 15
3 let str = "Saludo";
4
5 function ejemplo(){
6     str="Ejemplo";// PyTypeCraft hace referencia a la variable global str
7     let x = 0; // PyTypeCraft crea una nueva variable local
8     for (let i = 0; i < 10; i++){
9         let x: number;// Creando variable local.
10        x = i * 2; // Gracias a la variable 'local' no hace referencia a la variable de la línea 6
11        console.log(x);// Imprime: 2 4 6 8 10...
12    }
13    console.log(x);// 0 --> la variable nunca fue modificada
14 }
15
16 ejemplo();
17 console.log(x);// 15
18 console.log(str);// Ejemplo --> Modificada dentro de ejemplo()
```

```
1 // Ejemplo 2: Aclaraciones de alcance de variables
2 let x = 15;
3 let y = 44;
4
5 function ejemplo2() {
6     y = 5; // Se modifica la variable global
7     console.log(x); // Primero busca en el entorno local y luego en el global
8 };
9
10 ejemplo2();
11
12 console.log(x);
13 console.log(y);
```

```
1 let x = 3;
2
3 function ejemplo3() {
4     for (let i = 0; i < 5; i++) {
5         let x = null; // Aquí 'x' es local para el bloque del bucle for y está inicializado con 'null'.
6         console.log(x); // Esto imprimirá 'null' en cada iteración del bucle.
7     }
8 }
9
10 ejemplo3();
```

**Miércoles 13 de junio hasta las 23:59 P.M.**

## 4.6. Funciones

### 4.6.1. Creación de funciones

Las funciones en PyTypeCraft se crean con la palabra clave *function* seguida del nombre de la función y, entre paréntesis, los parámetros de entrada de la función. En PyTypeCraft es obligatorio utilizar la instrucción *return* para retornar un valor. En caso no se utilice o se utilice *return* sin valor, la función devolverá nada (el dato *null*).

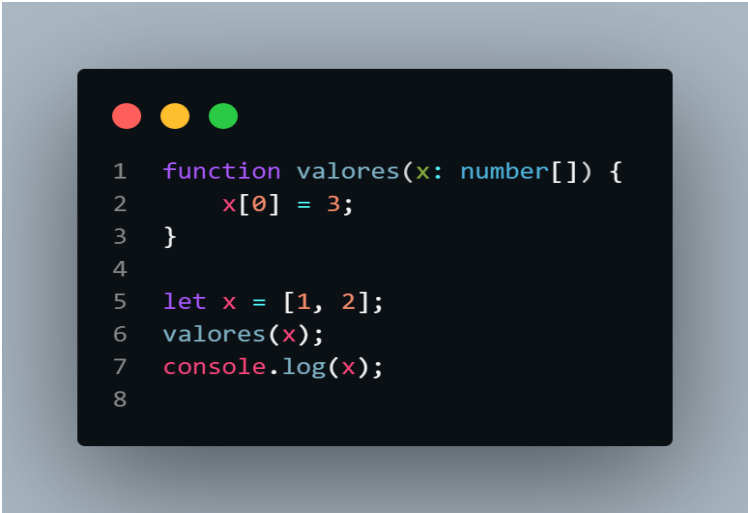
```
function NOMBRE_FUNCION (LISTA_PARAMETROS) {  
  LISTA_INSTRUCCIONES  
};
```

### 4.6.2. Llamada a funciones

La llamada a funciones se realiza con el nombre de la función, y entre paréntesis, los parámetros a pasar.

### 4.6.3. Paso por valor o por referencia

En PyTypeCraft, los únicos tipos que son pasados por referencia son los arreglos y *struct*, por lo que si se modifican dentro de una función también se modificarán fuera. El resto de tipos son pasados por valor.



```
1  function valores(x: number[]) {  
2    x[0] = 3;  
3  }  
4  
5  let x = [1, 2];  
6  valores(x);  
7  console.log(x);  
8
```

## 4.7. Condicionales

El lenguaje PyTypeCraft cuenta con sentencias condicionales, la evaluación condicional permite que porciones de código se evalúen o no se evalúan dependiendo del valor de una expresión booleana. Estos se definen por las instrucciones *if*, *elseif*, *else*.

**Miércoles 13 de junio hasta las 23:59 P.M.**

Consideraciones:

- Las instrucciones *else if* y *else* son opcionales.
- La instrucción *else if* se puede utilizar tantas veces como se desee.

```
1  let x = 8; // Definimos un valor para 'x', se necesitará para ejecutar el código
2
3  // Instrucción if
4  if (x === 8) {
5      let var1 = x + 8;
6      console.log(var1.toString());
7  }
8
9  // Instrucción if, elseif, else
10 if (x === 8) {
11     let var1 = x + 8;
12     console.log(var1.toString());
13 } else if (x < 8) {
14     let var1 = x / 3;
15     console.log(var1.toString());
16 } else {
17     console.log("Error");
18 }
19
20 // Instrucción if, else
21 if (x === 10) {
22     let var2 = x + 10;
23     console.log(var2.toString());
24 } else {
25     console.log(x+8);
26 }
```

## 4.8. Loops

En el lenguaje PyTypeCraft existen dos sentencias iterativas, este tipo de sentencias son aquellas que incluyen un bucle sobre una condición, las sentencias iterativas que soporta el lenguaje son las siguientes:

### 4.8.1. Ciclo while

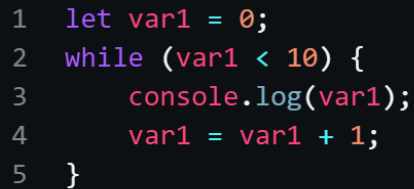
Esta sentencia ejecutará todo el bloque de sentencias solamente si la condición es verdadera, de lo contrario las instrucciones dentro del bloque no se ejecutarán, seguirá su flujo secuencial.

Consideraciones:

- Si la condición es falsa, detendrá la ejecución de las sentencias de la lista de instrucciones.
- Si la condición es verdadera, ejecuta todas las sentencias de su lista de instrucciones.

**Miércoles 13 de junio hasta las 23:59 P.M.**





```

1  let var1 = 0;
2  while (var1 < 10) {
3      console.log(var1);
4      var1 = var1 + 1;
5  }

```

### 4.8.2. Ciclo for

Esta sentencia puede iterar sobre un rango de expresiones, cadena de caracteres (*"string"*) o arreglos. Permite iniciar con una variable como variable de control en donde se verifica la condición en cada iteración, luego se deberá actualizar la variable en cada iteración.

Consideraciones:

- Contiene una variable declarativa que se establece como una variable de control, esta variable servirá para contener el valor de la iteración.
- La expresión que evaluará en cada iteración es de tipo rango, string o array. Aunque también se puede especificar mediante una variable.



```

1  // Recorre rango de 1:4
2  for (let i = 1; i <= 4; i++) {
3      console.log(i, " ");
4  }
5
6  // Recorre las letras de la cadena
7  for (let letra of "Hola Mundo!") {
8      console.log(letra, "-");
9  }
10
11 let cadena = "OLC2";
12 for (let letra of cadena) {
13     console.log(letra, "-");
14 }
15
16 for (let animal of ["perro", "gato", "tortuga"]) {
17     console.log(`${animal} es mi favorito`);
18 }
19
20 let arr = [1,2,3,4,5];
21 for (let i = 1; i < 4; i++) { // los índices en JavaScript y TypeScript comienzan en 0, no en 1
22     c

```

**Miércoles 13 de junio hasta las 23:59 P.M.**

### 4.8.3. Sentencias de transferencia

A veces es conveniente terminar un ciclo antes de que la condición sea falsa o detener la iteración de una sentencia loop antes de que se alcance el final del objeto iterable, además también es conveniente saltar unas sentencias de un ciclo en determinadas ocasiones y por para las funciones es necesario el retorno de un valor.

- Break
- Continue
- Return

Consideraciones:

- Se debe validar que la sentencia *break* y *continue* se encuentre únicamente dentro de una sentencia loop.
- Es necesario validar que la sentencia *break* detenga las sentencias asociadas a una sentencia loop.
- Es necesario validar que la sentencia *continue* salte a la siguiente iteración asociada a su sentencia loop.
- Es requerido validar que la sentencia *return* esté contenida únicamente en una función.

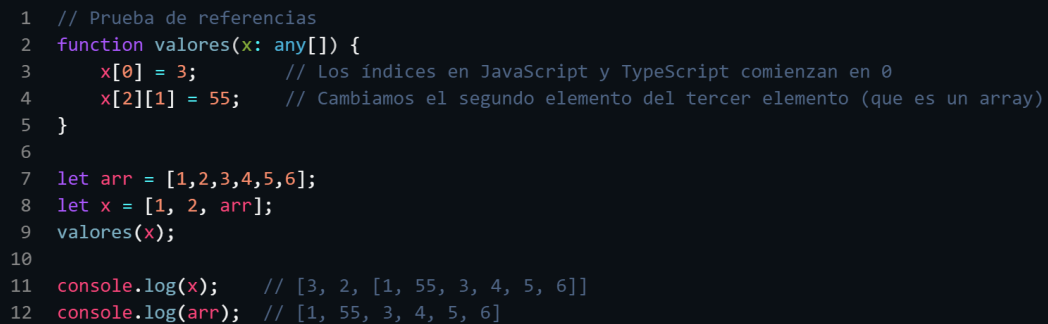
```
1 // Ejemplo break
2 while (true) {
3     console.log(true); // Imprime true solo una vez
4     break;
5 }
6
7 // Ejemplo continue
8 let num = 0;
9 while (num < 10) {
10     num = num + 1;
11     if (num === 5) {
12         continue;
13     }
14     console.log(num); // Imprime 1 2 3 4 6 7 8 9 10
15 }
16
17 // Ejemplo de return
18 function funcion() {
19     let num = 0;
20     while (num < 10) {
21         num = num + 1;
22         if (num === 5) {
23             return 5;
24         }
25         console.log(num);
26     }
27     return 0;
28 }
29
30 console.log(funcion()); // Para ver el valor que retorna la función
31
```

**Miércoles 13 de junio hasta las 23:59 P.M.**

## 4.9. Arreglos

En PyTypeCraft, se cuenta este tipo de dato compuesto y mutable. Puede contener cualquier tipo de dato.

Además, toma en cuenta que al tratarse de un tipo mutable, maneja referencias, por ejemplo:



```
1 // Prueba de referencias
2 function valores(x: any[]) {
3     x[0] = 3;        // Los índices en JavaScript y TypeScript comienzan en 0
4     x[2][1] = 55;    // Cambiamos el segundo elemento del tercer elemento (que es un array)
5 }
6
7 let arr = [1,2,3,4,5,6];
8 let x = [1, 2, arr];
9 valores(x);
10
11 console.log(x);    // [3, 2, [1, 55, 3, 4, 5, 6]]
12 console.log(arr);  // [1, 55, 3, 4, 5, 6]
```

## 4.10. Structs

Los *structs* son tipos compuestos que se denominan registros, los tipos compuestos se introducen con la palabra clave *interface* seguida de un bloque de nombres de campos, opcionalmente con tipos usando el operador “.”.

Consideraciones:

- Los atributos sin especificar el tipo, en consecuencia pueden contener cualquier tipo de valor.
- Los *structs* también se pueden utilizar como retorno de una función.
- Las declaraciones de los *structs* se pueden utilizar como expresiones.
- Los atributos se pueden acceder por medio de la notación “.”.



```
1  interface Carro {
2      placa: string;
3      color: string;
4      tipo: string;
5  }
6
7  let c1: Carro = {
8      placa: "090PLO",
9      color: "gris",
10     tipo: "mecanico"
11 };
12
13 let c2: Carro = {
14     placa: "P0S921",
15     color: "verde",
16     tipo: "automatico"
17 };
18
19 // Asignación Atributos
20 c1.color = "cafe";    // Cambio aceptado
21 c2.color = "rojo";   // Cambio aceptado
22
23 // Acceso Atributo
24 console.log(c1.color); // Imprime cafe
```

## 4.11. Tabla de Resumen de instrucciones

Instrucción	Descripción
console.log	Imprime el resultado de la expresión agregando el salto de línea.
Declaración y asignación	Asigna el resultado de la expresión a una variable existente, si no existe la variable será creada en esta instrucción.
Llamada a funciones	Llama a una función para que sea ejecutada, luego regresa al flujo normal donde se realizó la llamada.
Function	Son secuencias de instrucciones definidas en un bloque para ser ejecutadas en una llamada de la función.
If	La instrucción condicional ejecuta el bloque de instrucciones si la condición es verdadera.
Else If	Si la condición en if es falsa, esta instrucción evaluará una nueva condición si es verdadera ejecuta las instrucciones del bloque.
Else	Si las condiciones en if y elseif son falsas se ejecuta las instrucciones que estén en else.
While	Esta instrucción ejecuta el bloque de instrucciones si la condición es verdadera.
For	La instrucción for puede iterar sobre tipos iterables como lo son rangos, arreglos y cadenas.
Break	Esta instrucción termina la ejecución de una instrucción cíclica antes de que se alcance el final del objeto iterable.
Continue	Esta instrucción salta a la siguiente iteración sin ejecutar las instrucciones restantes.
Return	La instrucción return se utiliza para devolver un resultado en las funciones.

## 5. Reportes generales

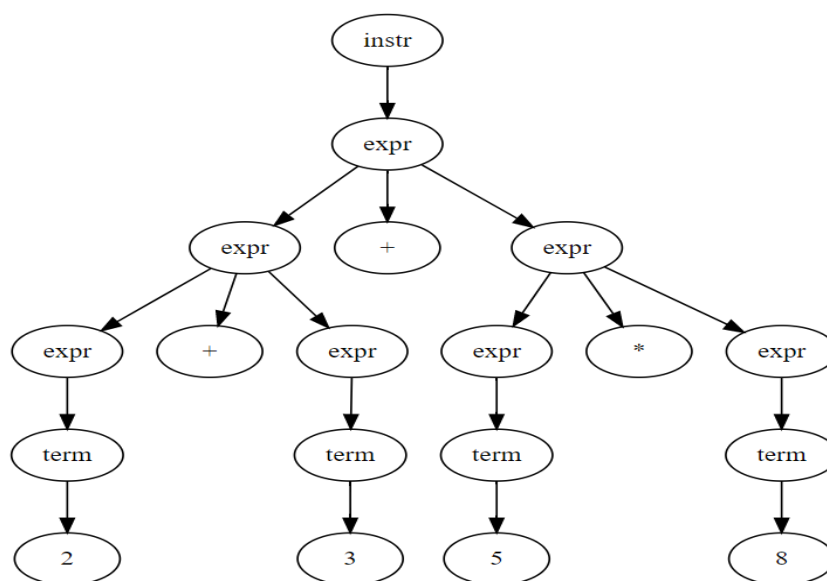
### 5.1. Tabla de Símbolos

En este reporte se solicita mostrar la tabla de símbolos después de la ejecución del archivo. Se deberán mostrar todas las variables, funciones y struct reconocidas, junto con su tipo y toda la información que el estudiante considere necesaria. Este reporte al menos debe contener la fila y columna de la declaración del símbolo junto con su nombre, tipo y ámbito. En el caso de las funciones, deberá mostrar el nombre de sus parámetros, en caso tenga.

Nombre	Tipo	Ámbito	Fila	Columna
x		valores	2	18
valores	Funcion	Global	2	1
arr	arreglo	Global	6	1
x	arreglo	Global	7	1

### 5.2. Árbol de análisis sintáctico

Deberá mostrar el árbol de análisis sintáctico que se generó al analizar el archivo de entrada de acuerdo con su gramática. El cual es un árbol con las siguientes propiedades: a) la raíz se etiqueta con el símbolo inicial. b) Cada hoja se etiqueta con un terminal, o con  $\epsilon$ . c) Cada nodo interior se etiqueta con un no terminal.



**Miércoles 13 de junio hasta las 23:59 P.M.**

### 5.3. Tabla de errores

Su aplicación deberá ser capaz de detectar y reportar todos los errores semánticos que se encuentren durante la ejecución. Su reporte debe contener como mínimo la siguiente información.

- Descripción del error.
- Número de línea donde se encontró el error.
- Número de columna donde se encontró el error.
- Fecha y hora en el momento que se produce un error.

No.	Descripción	Línea	Columna	Fecha y hora
1	El struct Persona no fue declarado	112	15	14/8/2021 20:16
2	El tipo string no puede multiplicarse con un real	80	10	14/8/2021 20:16
3	No se esperaba que la instrucción break estuviera fuera de un ciclo.	1000	5	14/8/2021 20:16

## Entregables y Calificación

Para el desarrollo del proyecto se deberá utilizar un repositorio de GitHub, este repositorio deberá ser privado y tener a los auxiliares como colaboradores.

### 5.4. Entregables

El código fuente del proyecto se maneja en GitHub por lo tanto, el estudiante es el único responsable de mantener actualizado dicho repositorio hasta la fecha de entrega, si se hacen más commits luego de la fecha y hora indicadas no se tendrá derecho a calificación.

- Código fuente y archivos de compilación publicados en un repositorio de GitHub cada uno en una carpeta independiente.
- Enlace al repositorio y permiso al auxiliar para acceder. Para darle permiso al auxiliar, agregar este usuario al repositorio
  - DiiAns23
- Aplicación web con la funcionalidad del proyecto publicada en AWS.

**Miércoles 13 de junio hasta las 23:59 P.M.**

## 5.5. Restricciones

- La herramienta para generar los analizadores del proyecto será Python PLY. La documentación se encuentra en el siguiente enlace <https://www.dabeaz.com/ply/>.
- No está permitido compartir código con ningún estudiante. Las copias parciales o totales tendrán una nota de 0 puntos y los responsables serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas.
- El resultado final del proyecto debe ser una aplicación web funcionando en AWS, no será permitido descargar el repositorio y calificar localmente.
- El desarrollo y entrega del proyecto es en parejas.

## 5.6. Consideraciones

- Es válido el uso de cualquier Framework para el desarrollo de la aplicación siempre y cuando la aplicación final pueda ser publicada en AWS.
- El repositorio únicamente debe contener el código fuente empleado para el desarrollo, no deben existir archivos PDF o DOCX.
- El sistema operativo a utilizar es libre.
- Se van a publicar archivos de prueba y sintaxis del lenguaje en el siguiente repositorio.
- El lenguaje está basado en Typescript, por lo que el estudiante es libre de realizar archivos de prueba en esta herramienta, el funcionamiento debería ser el mismo y limitado a lo descrito en este enunciado.

**Miércoles 13 de junio hasta las 23:59 P.M.**



## 5.7. Calificación

- La calificación se realizará dentro de la máquina de los auxiliares, ya que es muy importante que tengan la última versión de su proyecto subida a Heroku y las rutas definidas anteriormente.
- Se probará que el estudiante genere el compilado correcto y que esté siendo ejecutado en AWS.
- Durante la calificación se realizarán preguntas sobre el código y reportes generados para verificar la autoría de este, de no responder correctamente la mayoría de las preguntas se reportará como copia.
- Se tendrá un máximo de 45 minutos por estudiante para calificar el proyecto.
- La hoja de calificación describe cada aspecto a calificar, por lo tanto, si la funcionalidad a calificar falla en la sección indicada se tendrá 0 puntos en esa funcionalidad y esa nota no podrá cambiar si dicha funcionalidad funciona en otra sección.
- Si una función del programa ya ha sido calificada, esta no puede ser penalizada si en otra sección la función falla o es errónea.
- Los archivos de entrada permitidos en la calificación son únicamente los archivos de pruebas preparados por los tutores.
- Los archivos de entrada podrán ser modificados solamente antes de iniciar la calificación eliminando funcionalidades que el estudiante indique que no desarrolló.
- Los archivos de entrada podrán ser modificados si contienen errores semánticos no descritos en el enunciado o provocados para verificar el manejo y recuperación de errores.

## 5.8. Entrega de proyecto

- La entrega será mediante GitHub, y se va a tomar como entrega el código fuente publicado en el repositorio a la fecha y hora establecidos.
- Cualquier commit luego de la fecha y hora establecidas invalidará el proyecto, por lo que se calificará hasta el último commit dentro de la fecha válida.
- Fecha de entrega:

**Miércoles 13 de junio hasta las 23:59 P.M.**