LCON

LENGUAJE LCON

LENGUAJE DE PROGRAMACIÓN

ILMER CONDOR

2025

CAPITULO I

I. INTRODUCCIÓN

a. Aprende Python aprendiendo LCON

Mientras vaya aprendiendo el lenguaje LCON, Usted estará aprendiendo el Lenguaje Python.

LCON es muy fácil. Y sus instrucciones están en español; es decir, usted codificará un programa fuente en LCON como si estuviera escribiendo órdenes para que las ejecute otra persona que sabe e interpreta español.

Pero, mientras tanto, y sin saberlo estará aprendiendo Python, desde cero.

El Lenguaje LCON permite es interpretado hacia el lenguaje Python a través del compilador **IceCompilador01** también llamado traductor quien cheque la sintaxis del programa fuente escrito en el lenguaje LCON y los ejecuta línea por línea realizando dos tareas:

Convertir el programa fuente al Python y ejecutando la orden dada en las instrucciones de dicho programa fuente.

En lo sucesivo al IceCompilador01 lo llamaremos simplemente compilador.

Finalmente, aprendiendo LCON usted estará aprendiendo a:
- Programar
- El lenguaje LCON

- El lenguaje Python

El IceCompilador01 es más que un intérprete como el antiguo Basic o algunos actuales que chequean la sintaxis del programa fuente y, si no tiene errores, lo ejecuta.

Chequea la sintaxis, la definición (aunque no es obligatoria) y uso de las variables, genera un código en otro lenguaje (lenguaje del Python) y luego las ejecuta. Ejecuta el código creado en Python.

Desde ese punto de vista, IceCompilador es quien al ejecutarse procesa el programa fuente codificado en el lenguaje LCON generando la versión del mismo en Python y ejecutando línea por línea las instrucciones de dicho programa fuente.

Esta es la razón por la cual toda persona que desee usar el lenguaje LCON debe tener:

- instalado el Python.
- Instalado algunas librerías del Python

El procedimiento de la instalación del Python y sus librerías se expone en la sección 4 del Anexo.

b. ¿Qué es un lenguaje de programación? ¿Qué es el lenguaje LCON?

Un lenguaje es el medio que usamos los humanos para comunicarnos.

Para un mejor entendimiento entre ellos, todos los lenguajes presentan una estructura gramatical cuya sintaxis debe ser respetada para evitar otras interpretaciones.

Y para que el humano pueda comunicarse con una computadora también existen los leguajes de programación. También presentan una estructura gramatical que, dependiendo de su nivel de comunicación pueden tener una sintaxis rígida o libre. Como ejemplo tenemos el lenguaje Assembler y el Python; el primero con una estructura rígida y el segundo una gramática bastante libre y versátil.

En la actualidad el ser humano tiene grandes facilidades para aprender a programar pues estos lenguajes, a parte de su versatilidad y sencillez, podemos usar la Inteligencia Artificial (IA) que nos permite contar con una sugerencia en línea durante la programación de una aplicación o de un ejemplo de aprendizaje.

Todos esto lenguajes utilizan el lenguaje inglés para codificar las órdenes que debemos darle a la computadora. Hay un lenguaje creado en 2015 que acepta programas fuentes en español pero que está orientado para un servidor fundamentalmente.

El lenguaje que vamos a describir en este manual tiene por nombre LCON. Todas sus instrucciones están en español. Usa al IceCompilador01 como primer intérprete que los traduce al Python como segundo intérprete y los ejecuta como tal. Como lo era el Basic y otros, usa un compilador escrito en Python que chequea la sintaxis del programa fuente línea por línea y las ejecuta si no hay errores de sintaxis.

Esto último es lo que hace un intérprete; pero el IceCompilador01 es algo más que in traductor o intérprete: Genera un programa fuente en el lenguaje del Python. Así como hacían los lenguajes que creaban un programa en Assembler.

El IceCompilador01 oculta mucho detalle de programación en LCON. Por ejemplo, para realizar un análisis de regresión lineal, es suficiente una línea para procesar: **Estad();** y otra para capturar el detalle de la emisión del análisis: **resultado = reg(Y,X);**.

Tiene como fundamento los principios y conceptos básicos como variables, tipos de datos, estructura de control y las funciones que son esenciales para instruir a la PC o dispositivos como los celulares en los que se puede instalar el Python, además de hacerlo en las tabletas o laptop.

Por su simplicidad y fácil de comprender, puede ser usado por toda persona que quiera aprender los otros lenguajes que le permitan crear aplicaciones como el Python, el Android en sus diferentes entornos, lenguajes en el entorno Visual y otros.

Del mismo modo, por su simplicidad y sencillez el trazado de gráficos estadísticos y curvas polinómicas y de superficie son realizadas de una manera muy sencilla.

Si pensamos en las aplicaciones matemáticas, el lenguaje permite acceder al compilador

c. Historia y evolución

Los años 80 pretendía crear un lenguaje ensamblador en idioma español para máquinas de 32 bit de palabra. Su lenguaje se llamaba Utpía.

Posteriormente diseñé un programa intérprete en el Pascal de Borland.

No tuve mayor opción de desarrollo quizás por mi total dedicación a la educación universitaria.

Fue después del 2015 cuando me jubilé opté por aprender Android, Kolin, Jet Pack Compose en entornos de Android con los cuales hice una diversidad de aplicaciones como celulares, tutores de Estadística, R, Python, etc.

Fuel el 2020 en que me decidí construir un compilador usando el entorno de Python. Este compilador, que no era realmente un verdadero compilador sino un intérprete que se encargaba de leer un archivo llamado programa fuente cuyas órdenes o instrucciones estaban en el idioma español. Como lo hace el Python u otros intérpretes chequea la sintaxis línea por línea. Si está correcto pasa a ejecutarlas.

La siguiente versión de este compilador, que es la que está disponible ahora, es la versión 1.0. Decidí aumentarle la opción que permite el diseño de gráficos estadísticos, gráficos de polinomios matemáticos y trigonométricos de una manera totalmente elemental.

Siendo un software que presenta simplicidad y sencillez en su aprendizaje, sólo procesa arreglos unidimensionales o vectores dando lugar al manejo de listas. Para los tipos de problemas que resuelve, no los necesita arreglos bidimensionales.

Sin embargo, permite resolver todo problema estadístico como la regresión lineal y multilineal.

d. Características principales

En este apartado vamos a mencionar algunas de las características que tiene este lenguaje:

a) Su distribución es libre. Esto significa que cualquiera puede obtenerlo de forma gratuita. El archivo se puede descargar y luego implementar en cualquier máquina que tenga instalado el Python. Del mismo modo puede ser copiado llevado a otra máquina sea ésta una PC, laptop, Tablet o celular en un entorno Android.

b) Es de código abierto. Puesto que se obtiene libremente, cualquiera puede entrar al código y modificarlo sin requerimiento de permiso. Esto significa que, bajo la política del Open Source, cualquiera que lo modifique puede hacer uso para obtener beneficios propios sin la autorización del propietario original.

c) Es un lenguaje de alto nivel. Como todos los lenguajes de la actualidad, el LCON es un lenguaje de alto nivel, esto significa que está diseñado para leer y escribir el código de los programas fuentes de forma sencilla y de fácil lectura del mismo. Comparado con los lenguajes de programación como el C++, Java, Android, etc. El LCON es relativamente simple en su aprendizaje, en lectura de ejemplos de programa fuente y mucho más cercano al que desea iniciarse en la programación de las computadoras

d) Es un lenguaje interpretado. Esto significa que usa al Python como lenguaje intermedio quien lo procesa línea por línea. Como es el caso del lenguaje Java Script que al lenguaje Java como procesador de su código fuente y que por su versatilidad, tamaño y facilidad para aprender a programar es preferido al lenguaje Java. Los mismo ocurre con el Kotlin o el Jet Pack Compose con el Android. Las estructuras del Kotlin y, más aún el Jet Pack Compose se acercan al nuevo aspirante a programador que hacerlo con el Android.

e) No es un compilador. Es decir, no chequea la sintaxis ni construye tablas de referencias para terminar en un lenguaje objeto binario o ensamblador. Traduce cada una de sus sentencias al lenguaje Python para luego ser interpretadas y ejecutadas en el entorno del Python.

   En el caso de los compiladores, estos tienen dos fases por las que pasan todos los programas fuentes codificados en el lenguaje reconocido por el compilador.

e. Instalación y entorno de trabajo

Para disponer del compilador llamado **IcedCompilador.1.py** debe descargarlo haciendo uso de la siguiente dirección:

https://www.icepag.com/IceCompilador01.py

lo debe guardar en una carpeta particular, donde también puede guardar todos sus archivos fuentes y los datos requeridos o disponerlos en carpetas y unidades diferentes.

En el momento de abrir el compilador notará que hay una ruta que permite el acceso a la carpeta y archivos fuentes. Usted debe actualizarlo de forma que use sus propias carpetas y unidades.

Para grabar el código de un programa fuente usando el lenguaje LCON sólo necesita de un editor de texto. El archivo fuente deberá guardarse en una carpeta sea de la PC u otro dispositivo de almacenamiento.

Para procesar dicho archivo debe tener instalado el Python igual o superior a la versión 3.13 así como las librerías os, math, numpy, matplotlib.pyplot. Los últimos tres deberán ser instalados usando la herramiento PIP en el Shell del Windows.

Toda vez que desee procesar un archivo fuente, deberá tener cargado el Python y, usando el comando FILE, cargar el compilador para, luego de ejecutarlo usando RUN, ingresar el archivo fuente.

El siguiente ejemplo muestra el procedimiento para ejecutar un archivo fuente:

Paso 1: Supongamos que se ha codificado un programa fuente usando un editor de texto. Su nombre: EjPrg00a.txt

```
Leer a;
Leer “b: “,b;
Imprimir a;
Imprimir “b = “,b;
```

Paso 2: Ejecutar el IceCompilador01 (que se supone ya está abierto)

Paso 3: Se le pedirá la ruta y nombre del archivo fuente. En el caso del ejemplo anterior: d:/pyCcon/EjPrg00a.txt

Paso 3: En el ejemplo anterior, hay dos instrucciones que leen datos desde la consola. Esto implica que se deberá digitar un dato para a y otro para b. A continuación, se debe visualizar el valor de las variables a y b.

CAPITULO II

II. FUNDAMENTOS DEL LENGUAJE

a. Sintaxis básica

Un programa, escrito en el lenguaje LCON, está compuesto por un conjunto de sentencias cada una de las cuales ocupan una línea.

La forma cómo se forma la sentencia es de formato libre. Es decir,
a = 12;
b = 20;
son válidos.

Como se puede ver, cada sentencia debe terminar en un punto y coma (;).

El programa escrito en el lenguaje LCON recibe el nombre de programa fuente.

Como en el caso del Lenguaje Java Script o el Visual Basic de Aplicaciones usado en el Excel, se tiene la libertad de definir o no cada una de las variables al inicio del programa.

Cada sentencia es construida por la combinación de variables y operadores. Las variables contienen los datos y estos son procesados por los operadores que pueden ser aritméticos o lógicos.

A continuación, presentamos un ejemplo codificado en dos formas diferentes y que, al procesarlo produce los mismos resultados.

Primera forma

```
Definir {
Var a: Entero;
 Var B:     Real;
}
       a =      10;
B = 15;
xSuma = a +      B;
Imprimir a;
Imprimir "B = ",B;
Imprimir "La suma es: ",    xSuma;
```

Segunda forma

```
a = 10;
B = 15;
xSuma = a+b;
Imprimir a;
Imprimir "B = ",B, " La suma es: ",xSuma;
```

Observaciones:
EL nombre de las variables puede ser cualquiera; de una o más letras. No hay diferencia en el uso de mayúscula o mayúscula. Puede contener dígitos pero no caracteres especiales como coma, (,), *, etc.
Cada sentencia termina en punto y coma.
En ambos casos no se ha definido xSuma. Pero habrá algunos casos en los que sí será necesario definirlos.

Por ejemplo, cuando se trate de leer un dato, la variable que reciba el dato debe estar previamente definida de otra forma se tendrá error de sintaxis durante la compilación.
Por formalidad se debiera definir todas las variables.

Si hubiera una o más líneas cuyo contenido estuviera precedida por el carácter "#", la línea no será considerada pues esta será tomada como un comentario.

Si estuviera en blanco, la línea no se considera.

b. Procedimiento para compilar un programa fuente

Para el efecto tomemos el siguiente ejemplo:
Hemos codificado un programa y lo hemos grabado con el nombre EjPrg01.txt, en la carpeta pyCcon de la unidad D.

```
Definir {
X, Y: Vector;
}
a = 5;
b = -25;
xPago = 1250;
xTotalProducto = xPago + a*b
```

Ejecutamos el Python.

Desde el IDLE, abrimos el compilador (IceCompilador01.py).
Lo ejecutamos usando <RUN Module>.
Cuando en la pantalla del IDLE se emita "Nombre del programa fuente, incluya la ruta: ", debemos digitar, en mi caso, d:/pyCcon/EjPrg01.txt y presionar <Intro>.

Esto es lo que haremos toda vez que se requiere compilar un programa fuente; incluso si se desea procesar nuevos datos.

Respecto a este programa fuente, al ingresar el nombre del archivo fuente, tendremos un error en la pantalla en texto de color rojo, como el siguiente:

```
>>>
========== RESTART: D:\pyCon\IceCompilador01.py ========
Nombre del programa fuente, incluya la ruta: d:/pyCcon/EjPrg01.txt
Traceback (most recent call last):
  File "D:\pyCon\IceCompilador01.py", line 798, in <module>
    obj = compile(cad,"ice01","exec")
  File "ice01", line 4
    X, Y: Vecto=X, Y: Vector
       ^
SyntaxError: only single target (not tuple) can be annotated
>>>
```

Allí podemos apreciar que en "X, Y: Vector" se presenta un error. En efecto, al definirse las variables, se debe anteceder con la palabra reservada "Var". Esto implica que, usando el editor abrimos el archivo fuente y procedemos a corregirlo.

Al repetir el procedimiento, encontramos otro error de compilación:

```
>>>
========== RESTART: D:\pyCon\IceCompilador01.py ========
Nombre del programa fuente, incluya la ruta: d:/pyCcon/EjPrg01.txt
Traceback (most recent call last):
  File "D:\pyCon\IceCompilador01.py", line 798, in <module>
    obj = compile(cad,"ice01","exec")
  File "ice01", line 7
    xTotalProducto = xPago + a*
                             ^
SyntaxError: invalid syntax
>>>
```

En este caso el error se presenta en la línea 7 después de "a*". En efecto, en el archivo fuente observamos que la sentencia debe terminar con ";".

Si luego de corregir este error lo volvemos a compilar veremos que no se emite ningún error compilación.

Esto indica que el programa fuente está libre de errores de compilación. Aunque tampoco emite resultado alguno.

¿Por qué no vemos resultado alguno?

Falta la sentencia de impresión.

c. Tipos de datos

LCON reconoce los siguientes tipos de datos:

- Numéricos
  Datos numéricos Enteros
  Datos numéricos Reales

- Cadenas
  Secuencia de caracteres

- Listas
  Tipo de dato que almacena un conjunto de datos de tipo numérico o cadena. Están separados por comas y encerrados entre corchetes.

  Por ejemplo:
  x = [120,80,50,100,90, 150]
  diaSem = [‘Lun’, ’Mar’, ’Mie’, ’Jue’, ’Vie’, ’Sab’, ’Dom’]
  secta = [125, ‘Selena’,3.141592,-250.45,’Mrgot’,’Ok’]

- Arreglos unidimensionales
  Los arreglos unidimensionales son los vectores. En Python tienen una estrecha relación con las listas pues dada una lista numérica, esta puede ser convertida en un arreglo para propósitos de usarlo en operaciones matemáticas vectoriales.

Observación importante:

Cuando se trata de almacenar o guardar un conjunto de datos en un arreglo, se usa la variable **aX** que, va a constituir una palabra reservada.

d. Variables

Como hemos dicho líneas arriba, un programa fuente se construye mediante el uso de las variables.

Todas las operaciones que se realizan en un programa que usa un lenguaje de alto nivel, usa variables que contienen los datos que deben ser procesados.

Para ello las variables tienen un nombre única forma de referirnos a ellas.

El nombre de una variable es cualquier combinación de caracteres literales en mayúscula o minúscula a las cuales se pueden añadir uno o más dígitos.

Por ejemplo:

A, c, nro, xTit, nro1, nro2, prUnit, etc.

Hay, sin embargo, nombres que no se pueden usar o letras con las que no pueden comenzar el nombre de una variable.

- xA: Está reservada para almacenar uno o varios datos formando una cadena de caracteres.
- El nombre de una variable no puede empezar con OIxxx ni tampoco TGxxx. TGuardar y OImprimir son instrucciones que permiten el uso del arreglo xA tanto para guardar en él como para imprimir su contenido.

Por formalismo y en algunos casos por necesidad, las variables deben ser declaradas o definidas al inicio del programa.

Una variable puede ser usada para una operación y luego para otra, siempre conservando su tipo de dato.

e. Operadores

Puesto que para resolver un problema se requiere de operadores matemáticos y lógicos y de comparación, el lenguaje LCON usa los siguientes operadores:

a) Aritméticos

Nombre	Operador	Ejemplo
Potencia	**	a**b
División	/	a/b

Multiplicación	*	a*b
Resta	-	a-b
Suma	+	a+b

b) Relacionales o de comparación

Permiten relacionar o comparar el contenido de una variable o expresión con otra o una constante.

Nombre	Operador	Ejemplo
Igual a	==	a = b
Diferente	¡=	a ¡= b
Mayor	>	a > b
Menor	<	a < b
Mayor o igual	>=	a >= b
Menor o igual	<=	a <= b

Nombre	Operador	Ejemplo
Y	and	m and n
O	or	m or n

f. Listas

Conjunto de elementos separados por una coma y encerrados entre paréntesis.

Ejemplos

i) X = [12,17,13,08,16,15,14,14,06,16]
ii) saldo = [12650,726345,230567,10987,0,27262]
iii) meses = [“Enero”, “Febrero”, “Marzo”, “Abril”, “Mayo”, “Junio”, “Julio”, “Agosto”, “Setiembre”, ”Octubre”, “Noviembre”, “Diciembre”]
iv) Prod = [“Azúcar blanca”,1200,5.40,0.05]

CAPITULO III

III. ENTRADA Y SALIDA

a. Lectura de datos

El lenguaje LCON tiene a la instrucción ***input …*** como método de ingreso de datos.

Sintaxis:

Leer [mensaje1 ,] variable1;

Interpretación:

Leerá el dato que será almacenada en “variable1”.

En esta sintaxis el uso de los corchetes indica que su contenido es opcional. En el caso de esta sentencia cuando no se coloca el “mensaje”, el compilador emite el siguiente mensaje: “Ingresa el dato: “.
“Variable1” debe estar definida previamente.

Ejemplos:

Leer “Ingresa un número: “, nro;

Leer “Nombre del producto: “, nomProd;

Ejemplo 01

Definir {
Var m, n: Entero;

```
}
Leer “Ingrese el valor para m: “,m;
Leer “Ingrese el valor para n: “, n;
suma = m + n;
prod = m*n;
resta = m – n;
cociente = m/n;
potencia = m**n;
```

Ejemplo 02

Supongamos que se trata de procesar los datos de n alumnos. Se desea codificar la lectura de los datos.

Variables a ser usadas:

xCod: Código de alumno
xNom: Nombre del alumno
edad
Monto: Monto de dinero que tiene

El programa es el siguiente:

```
Definir {
Var xCod, xNom: Cadena;
Var n, edad: Entero;
Var Monto: Real;
}
Leer "Nro de alumnos a procesar: ",n;
Leer "Codigo: ", xCod;
Leer "Nombre: ",xNom;
Leer "Edad: ",edad;
Leer "Monto que posee: ", Monto;
```

b. Impresión de datos/resultados

Los resultados que puedan obtenerse al compilar y ejecutar un programa fuente en LCON, se reciben en la pantalla de su equipo.

La instrucción que nos permite hacer esto es la instrucción ***Imprimir …***

Sintaxis:

Imprimir [mensaje1 ,] variable 1 [[, mensaje2] , variable 2]

Interpretación:

Permite imprimir en el IDL el valor de una o dos variables.
Los corchetes indican que su contenido es opcional; es decir, se pueden dejar de usar.

Aquí varias formas de uso de esta instrucción:

Imprimir a;

Imprimir “a = “,a;

Imprimir a,b;

Imprimir a, “ , “, b;

Imprimir “valor de a: “, a, “valor de b = “, b;

Imprimir “valor de a = “,a, “\nvalor de b = “, b;

El uso de “\n” antes del mensaje2 permite imprimir el resto de la sentencia en la siguiente línea; es decir, “\n” es el código para salto de línea.

A continuación, presentamos un ejemplo con varias modificaciones:

Ejemplo 03

```
Definir
  {
 Var msg: Cadena;
 Var a, b: Entero;
 Var c, d: Real;
 }

msg = "Segundo número: ";
Leer "Primer numero: ", a;
Leer "Segundo número: ", b;
c=a+b;
d=a*b;

# Se imprime los resultados
#
Imprimir "a = ",a, " b = ",b;
```

```
Imprimir "La suma: ", c;
Imprimir d;
Imprimir "La suma: ",c, "\nEl producto: ",d;
```

Grabemos este programa con el nombre **EjPrg02a.txt**

Explicación:

Se empieza definiendo las variables en la sección Definir { }. Observe que se ha definido a ***msg*** como una variable de cadena, se le ha asignado un valor, pero no ha sido usado.
Se leen valores numéricos enteros para ***a*** y ***b*** emitiendo un mensaje que sirve para orientar al usuario que está ejecutando el programa.

Se hace los cálculos para ***c*** y ***d***.
Hay una línea que empieza con "#" que no es procesada.

Luego tenemos cuatro formas diferentes de imprimir los datos y resultados: En el primer caso se imprime ***a*** y ***b*** con un mensaje previo. En el segundo caso sólo se imprime un mensaje y el valor de la suma en ***c***. En la tercera sólo se imprime el valor de ***d***, no hay mensaje explicativo de lo que imprime. Finalmente se imprime ***c*** y ***d*** con sus respectivos mensajes, pero no en la misma línea. En una línea se imprime un mensaje y el valor de la suma y, luego de realzar un salto de línea "\n", se imprime el valor de b.

Ejemplo 04

El siguiente programa usa la sentencia Imprimir de forma muy elemental, sin mensajes.

```
# Se lee dos números,
# Se calcula su suma y producto
# Se imprime el resultado

#Definir:
   {
  Var a: Entero;
  Var b: Real;
  Var c, d: Real;
Var u,v: Real;
 }
u = -10;
v = 100;
Leer " a = ",a;
Leer "  b = ", b;
```

```
c=a+b;
d=a*b;
e = u+v;
f = a+b+u+v;
# Se imprime los resultados
#
Imprimir c;
Imprimir d;
Imprimir e;
Imprimir f;
```

Guarde este programa con el nombre **EjProg02a.txt** y pase a ejecutarlo.

Como puede apreciar, la forma de imprimir los resultados no permite reconocer qué está imprimiendo.

OBSERVACIÓN IMPORTANTE

Como habrá notado al procesar los programas de los ejemplos 3 y 4, hay un segmento inicial que se emite antes de los resultados. Esas son las sentencias del programa traducidos al Python.

No comentaremos por ahora las tres primeras líneas de dicho código.

Como hemos dicho antes, cuando LCON procesa un programa fuente, primero lo compila chequeando su sintaxis. Esté bien o mal, por cada línea procesada, se genera una línea de código en Python.

En los dos ejemplos puede apreciar que la instrucción ***Leer*** se ha traducido a ***input()***. Lo mismo sucede con ***Imprimir*** donde su equivalente es ***print()***.

Al leer un valor para ***a***, LCON lo lee como entero pues así fue definido. La conversión lo convierte en

a=int(input(' a = '))

lo mismo ocurre con ***b***

b=float(input(' b = '))

que lo lee como flotante pues así fue definido por LCON.

Las líneas correspondiente a ***Imprimir*** las convierte a ***print()*** sin ningún cambio.

Las otras líneas no sufren ningún cambio,

Ejemplo 05

Veamos el siguiente ejemplo:

```
# Se lee dos números,
# Se calcula su suma y producto
# Se imprime el resultado

Definir {
  Var a,      b, Iceb, xtot: Entero;
Var c, d: Real;
Var m, n: Entero; }
Leer "Valor de a: ",a;
Leer "Ingresa un numero para b: ", b;
Imprimir "Valor de a = ",a, "Valor de b = ",b;

# Cálculos
c=a+b;
d=a*b;
e=a**2+ b**2;

# Se imprime los resultados
#
Imprimir c, d;
Imprimir "El valor de e es: ",e;
```

Grabe este programa con el nombre **EjPrg04.txt.**

En este ejemplo hay variables definidas y no tienen uso. El lenguaje LCON no emite ninguna alerta sobre esto y menos lo detecta la traducción a Python pues al no ser usadas, no tienen su equivalente.

CAPITULO IV

IV. FUNCIONES Y MÉTODOS

En este capítulo vamos a describir el uso de instrucciones la ejecución condicional de una o más sentencias.

Del mismo modo describiremos la ejecución de una o más sentencias en forma repetitiva siempre que se cumpla alguna condición.

a. Condicionales

La repetición condicional de una o más líneas de código se realiza con la instrucción ***Si***.

Sintaxis

Primera forma:

Si condición entonces Sentencia_0;

Segunda forma:

Si condición {
Sentencia_1;
[Sentencia_2:
…]
}

Tercera forma:

Si condición {
Sentencia_1;
…
}
Caso contrario {
Sentencia_1;
…
}

En todos los casos ***condición*** es una expresión que contiene una condición de comparación que puede ser aritmética o lógica.

Explicación:

En la primera forma:

Si la condición de comparación se cumple, se ejecuta Sentencia_0.

Ejemplos

Si a < b entonces c = a + b;

Si tDesc = 1 entonces Desc = 0.12;

Si tDesc ¡= entonces Desc = 0.15;

En la segunda forma:

Si la condición de comparación se cumple, se ejecutan todas las sentencias que estén incluidas entre "{" y "}".

Ejemplos

Si a < b {
Leer "n = ",n;
Leer "m = ",m;
Imprimir "n = ",n, "\nm = ",m;

}

En la tercera forma:
Si la condición de comparación se cumple, se ejecuta todas las sentencias que estén incluidas entre las llaves y si no se cumple, entonces se ejecutan todas las sentencias incluidas entre las llaves de "***Caso contrario***"

Ejemplos

```
Si a < b {
c = a*b;
d = 10*a+20*b;
Imprimir "c = ",c, " d = ",d;
Caso contrario {
c = a/b;
d = 10*b-10*a;
Imprimir "c = ", c, " d = ", d;
}
```

Ejemplo06

Escriba un programa en LCON que lea dos números y luego imprima la de ellos si el primero es menor que el segundo, en caso contrario, que imprima su producto.

```
Definir {
Var a, b: Entero;
Var r, s : Real;
}

r = 0;
s = 0;

Leer "a = ",a;
Leer "b = ",b;

Si a < b entonces r = a + b;
Si a > b entonces r = a * b;

Imprimir "Valor de r = ", r;
```

Grabe este programa con el nombre **EjSi01.txt**
Si al procesar este programa, se ingresa 10 para ***a*** y 25 para ***b***, se imprimirá su suma; es decir, 35. Si se digitara 30 para ***a*** y 25 para ***b,*** se imprimirá 250.

Ejemplo07

¿Qué modificación se haría si se desea usar la tercera forma?

Se debe cambiar las dos sentencias ***Si*** por lo siguiente:

```
Si a < b {
r = a + b;
Caso contrario {
r = a*b;
}
```

Introduzca este cambio en el programa **EjSi01.txt** y vuelva a ejecutarlo.

Ejemplo08

Obtener las raíces de una ecuación cuadrática $ax^2+bx+c = 0$. Se debe emitir un mensaje si dicha ecuación no tiene raíces reales.

El programa fuente es el siguiente:

```
# Obtención de las raíces de una ecuación
# cuadrática
Definir {
Var a, b, c: Entero;
Var r1, r2 : Real;
Var tempo: Real;
}
# Leemos los coeficientes de la ecuación
Leer "a = ",a;
Leer "b = ",b;
Leer "c = ",c;

tempo = b**2-4*a*c;
Si tempo < 0 {
Imprimir "No tiene raices reales...";
}
Caso contrario {
r1 = (-b + tempo**0.5)/(2*a);
r2 = (-b - tempo**0.5)/(2*a);
Imprimir "Sus raices son:";
Imprimir "----------------";
Imprimir "r1 = ",r1, " r2 = ",r2;
```

Grabe este programa con el nombre **EjSi02.txt** luego ejecútelo.

Sugerencia:

Pruébense con:
x^2-6x+9
x^2-6x+8
6x^2+6x-36

OBSERVACIÓN IMPORTANTE:

Antes de continuar con el ingreso de los datos, veamos un momento lo que ocurre con la instrucción ***Si*** del LCON que ha sido traducida la instrucción ***if*** del Python. Las llaves usadas en el LCON sirvieron para convertirlo en un punto al final de la línea y luego la siguiente línea empieza con una indentación o sangrías en el Python.

Esto quiere decir que al usar if en Python debemos tener cuidado en el uso de las indentaciones; lo que no es requerido en JavaScript o C++.

b. Repetitivas

Las instrucciones repetitivas en LCON se usan para ejecutar un conjunto de sentencias o líneas de código repetidamente hasta que se cumpla o mientras se cumple cierta condición de comparación.

Estas repeticiones pueden realizarse un número finito de veces o bajo cierta condición de comparación.

En LCON disponemos de dos instrucciones:

Repetir

Mientras

i) Instrucción Repetir

Permite repetir una o más sentencias. Para ello dispone de dos formas:

Primera forma:

```
Repetir variable veces {
Sentencia_1;
Sentencia_2;
…;
}
```

Segunda forma:

```
Repetir variando variable de valInicial hasta valFinal {
Sentencia_1;
Sentencia_2;
…;
}
```

Tercera forma:

```
Repetir variando variable de valInicial hasta valFinal inc valor {
Sentencia_1;
Sentencia_2;
…;
}
```

Donde ***variable***, ***valInicial*** y ***valFinal*** y ***valor*** son variables numéricas con un valor entero.

La primera forma repite variable veces las sentencias incluidas entre las llaves.

La segunda forma repite las sentencias incluidas entre las llaves para variable que toma valores desde valInicial hasta val Final.

La tercera forma permite repetir las sentencias variando a ***variable*** desde ***valInicial*** hasta ***valFinal*** incrementando cada repetición en ***inc*** unidades a ***variable***.

Ejemplos:

```
Repetir 12 veces {
…
}
```

```
Leer “Nro de veces: “,n;
Repetir n veces {
…;
}
```

```
Repetir variando x de a hasta b {
…;
}
```

```
Leer “Val inicial: “,vi;
Leer “Val final: “,vf;
Leer “Increm,: “,increm;
Repetir variando i de vi hasta vf inc increm {
…;
}
```

Ejemplo 09

En el siguiente programa usando repetir contamos cuántas veces se repite el bucle. Y en la segunda parte se usa el bucle para sumar los n primeros números.

```
# Ejemplo
Definir
{
Var n: Entero;
}
Leer "Nro de repeticiones: ",n;

# Vamos a contar el nro de repeticiones
sumTot = 0;
Repetir n veces {
sumTot=sumTot+1;
}
Imprimir "Nro de repeticiones ", sumTot;

#Ahora vamos a sumar los primeros n números
i = 0;
sumTot = 0;
Repetir n veces {
i = i+1;
sumTot = sumTot + i;
}
Imprimir "La suma es = ", sumTot;
```

Grabe el programa con el nombre **EjPrg06.txt** y luego pase a ejecutarlo.

No olvidar que el bloque de sentencias que deben repetirse están encerradas entre las llaves.

Ahora observa el programa traducido a Python que está líneas arriba:

La instrucción **Repetir** ha sido reemplazada por **for**. La sentencia for debe terminar en ":" y las líneas siguientes deben estar indentadas.

Como se puede ver, en ambos casos, el for termina en la línea que ya no tiene la indentación respectiva. Esto ocurre con la sentencia print() que como se ve, empieza al inicio de la siguiente línea.

Ejemplo 10

Veamos el siguiente programa que, va a leer el nombre de un alumno, leerá también el número de notas que tiene y, luego de leer las notas calculará el promedio y la varianza de las notas.

El programa es el siguiente:

```
# Ejemplo
Definir
{
Var m, n, nota: Entero;
Var nombre: Cadena;
Var s, s2: Entero;
Var prom, varianza: Real;
}

s = 0;
s2 = 0;
Leer "Nombre de alumno: ", nombre;
Leer "¿Cuantas notas tiene? ", m;

Repetir m veces {
Leer "Nota: ", nota;
s = s + nota;
s2 = s2 + nota*nota;
}
prom = s/m;
varianza = (s2-m*prom*prom)/(m-1);
Imprimir "Su promedio es: ", prom, " con una varianza = ", varianza;
```

Grábelo con el nombre **EjPrg06a.txt** y ejecútelo.

Ejemplo 11

Vamos a calcular el ingreso diario de una persona durante **n** días donde se aplica el 8% de descuento. Al final se debe imprimir el monto total obtenido en los **n** días.

Empezaremos definiendo las variables a usar. Luego se deberá leer el número de días a procesar para luego repetir el procedimiento n veces.

El programa fuente en LCON es el siguiente:

```
# Sumar los ingresos de una semana
Definir {
Var i, n: Entero;
Var monto: Real;
}
totIngreso = 0.0;
Leer "Nro de dias: ", n;
i = 1;
# Vamos a contar el nro de repeticiones
Repetir n veces {
Leer "Ingreso del dia: ", monto;
dscto = 0.08*monto;
ingDia = monto - dscto;
totIngreso = totIngreso + ingDia;
i = i+1;
}
Imprimir "Ingreso total: ", totIngreso;
```

Como se puede ver, al interior de la instrucción Repetir se puede usar cualquiera de las instrucciones del lenguaje.

Grabe el programa con el nombre **EjPrg06b.txt**. Luego ejecútelo.

Observación importante:

La instrucción ***Repetir*** del LCON se convierte en ***for*** en Python.

Si bien en el lenguaje Python se puede incluir cualquier otra instrucción, en particular el mismo for, en LCON por ser un lenguaje para principiantes, sólo se permite usar una sola de ellas.

Ejemplo 12

En el siguiente ejemplo veremos el uso de la instrucción **Repetir** en la construcción de una tabla de los **n** primeros números.

El siguiente es el programa fuente:

```
#Tabla de los primeros 20 números
Definir {
Var n: Entero;}
Leer "Tamaño de la tabla n = ", n;

i = 0;

Repetir n+1 veces {
x = i;
x2 = i * i;
x3 = i **3;
x4 = i ** 4;
xra = i ** (0.5);
xrc = i**(1/3);
cad = str(x)+"  "+str(x2)+"  "+str(x3)+"  "+ str(x4)+"  "+str(xra)+"
"+str(xrc);
Imprimir cad;
}
```

Ejemplo 13

Ahora veamos el uso de la instrucción ***Repetir variando ...*** . En el siguiente ejemplo se desea realizar algunas operaciones para valores de una variable desde un valor inicial hasta un valor final. Y luego se añade el uso de **Repetir** simple.

El programa es el siguiente:

```
# Ejemplo
Definir
{
Var a, b, c: Entero;
Var      d: Real;
Var s, t : Real;
}
Leer "Ingresa un numero: ", a;
Leer "Ingresa otro numero: ", b;
Leer "Valor de c: ", c;
s = 0;
t = 10;
Repetir variando i de a hasta b {
s = s + c* i;
```

```
t = t *b;
}
d = s + t;
Imprimir "Sumas = ", s;
Imprimir "Productos = ", t;
Imprimir "Valor de d: ", d;
s = 0;
Repetir c veces {
s=s+30;
}
Imprimir "Valor de s = ", s;
```

En este ejemplo, la variable **i** toma valores desde **a** hasta **b** que son leídos desde el teclado.

Grabe el programa con el nombre EjPrg06d.txt y luego ejecútelo. Los valores para a y b sean cercanos si desea comprobar los resultados.

Observe también cómo las sentencias en el Python quedan traducidas y lo más importante: Su fácil lectura gracias a la indentación.

Ejemplo 14:

Finalmente, veamos un programa donde se usa la instrucción ***Repetir*** con la característica del incremento; es decir, incrementar en un valor ***incr*** cada vez que se ejecuta el conjunto de sentencias incluidas en el ámbito de ***Repetir***.

He aquí el ejemplo:

```
# Ejemplo
Definir
{
Var a, b, c: Entero;
Var       d: Real;
Var s, t : Real;
Var inc: Entero;
}
Leer "Ingresa un numero: ", a;
Leer "Ingresa otro numero: ", b;
Leer "Valor de c: ", c;
s = 0;
t = 10;
#Si se desea un incremento de 2 se debe usar
```

```
Leer "Incremento: ", inc;
d = 0;
Repetir variando i de a hasta b incr inc {
Imprimir "Valor de i = ",i;
s = s + c*i;
t = t *b;
d = d+s/t*100;
}
e = s+t;
Imprimir "Sumas = ",s;
Imprimir "Productos = ",t;
Imprimir "Valor de e: ",e;
Imprimir "Valor de d = ",d;
```

En este ejemplo la variable ***i*** empieza con el valor ***a*** en la primera iteración; en cada nueva iteración su valor se incrementa en ***inc*** hasta llegar ***a*** ser mayor que ***b***, en cuyo caso termina las iteraciones y se continúa con la siguiente sentencia fuera del ámbito del ***Repite***.

Grabe el programa con el nombre **EjPrg06e.txt** y pase a ejecutarlo.

Observe el programa generado en Python,
Los valores a, b y inc son colocados en range(a,b,inc).
La sentencia LCON: **Repetir variando i de a hasta b incr inc {** se ha convertido en sentencia Python: **for i in range(a,b,inc):**

ii) Instrucción Mientras

Sintaxis

```
Mientras expresCondicional {
Sentencia_1;
Sentencia_2;
…;
}
```

Explicación:

Mientras la **expresCondicional** se cumple, se ejecuta todo el bloque de sentencias limitadas por las llaves.

Ejemplos:

```
Mientras x < 10 {
```

```
Leer “a = “, a;
x = x + 1;
}
```

Ejemplo 15

Tomemos el siguiente programa en la cual se van a ejecutar dos sentencias mientras la variable **i** sea menor o igual a 10.

```
# Programa usando Mientras
Definir {
Var a, b: Entero;
Var c: Real;
Var i: Entero;
Var s, t: Real;
}
s=0;
Leer "a = ", a;
Leer "b = ", b;
i=0;
t = 0;
Mientras i<=10 {
s=s+a*i;
t = t + b*i/10;
i = i+1;
}
Imprimir "Nro de iteraciones: ", i,"\ns = ", s;
Imprimir "t = ", t;
```

En este ejemplo, **expresCondicional** es **i<=10**.
Esto significa que, previo al uso de la instrucción Mientras i debe tener un valor. Y, además, la variable de control, i, debe incrementar su valor hasta que llegue a ser superior a 10, en cuyo caso terminan las iteraciones y se continúa con la siguiente instrucción que está después de la llave “ } “.

Guarde este programa con el nombre **EjPrg07.txt** y ejecútelo.

Observación:

Al observar el programa en Python, líneas arriba, se verá que **Mientras** del LCON se convierte en **while** del Python que, como la instrucción for, también está indentada todas las sentencias incluidas entre “ { “ y “ } “.

Ejemplo 16

Veamos el siguiente ejemplo:

```
# Programa usando Mientras
Definir {
Var a, b: Entero;
Var c: Real;
Var i, j: Entero;
Var s, t: Real;
}
s=0;
Leer "a = ",a;
Leer "b = ", b;
i=0;
t = 0;
Mientras i<=10 {
s=s+a*i;
t = t + b*i/10;
i = i+1;
}
Imprimir "Nro de iteraciones: ", i,"\ns = ",s;
Imprimir "t = ",t;

j = 9;
s = 0;
Mientras j > a and j <= b {
Leer "c = ",c;
s = s + c;
Imprimir "j = ", j, "c = ",c;
j = j + 1;
}
Imprimir "Suma = ", s;
```

Guárdelo con el nombre **EjPrg07a.txt** y ejecútelo.

En la primera instrucción ***Mientras*** es como en el ejemplo anterior.

En el segundo: Si el valor leído para a es 10 y para b = 15, la expresión de comparación es “j > a and j <= b”. En este caso, la primera parte no se cumple; y aunque la segunda se cumpla, la expresión lógica no es verdadera; por lo cual, no se ejecuta todo el bloque.

Observación:

Si, con los mismos valores de a, b y j, cambiamos and por or, al comienzo no se cumple j>a pero sí j<=b; por tanto se ejecuta. Conforme j aumenta su valor, la primera condición se cumple, pero la segunda ya no; sin embargo, la expresión or sí se cumple; luego sigue ejecutando. Ahora seguirá ejecutando pues la primera se cumple pero no la segunda; pero al ser or, siempre seguirá ejecutando.

Cuando vemos que ocurre algo parecido (loop infinito), debemos cancelarlo usando <CTRL>+c

Ejemplo 17

En el siguiente ejemplo vamos a utilizar instrucciones ***Repetir*** y ***Mientras***, con las mismas variables, pero inicializadas adecuadamente.

```
# Programa usando Repetir y Mientras
Definir
{
Var a, b: Entero;
Var c: Real;
Var i: Entero;
Var s, t: Real;
}
s=0;
Leer "a = ",a;
Leer "b = ", b;
t=0;
Repetir variando x de a hasta b {
s = s+x*b;
t = t+ x*a;
}
Imprimir "Valor de s = ",s;
Imprimir "Valor de t = ", t;

i=0;
s=0;
t = 0;
Mientras i<=10 {
s=s+a*i;
t = t + b*i/10;
i = i+1;
}
Imprimir "Nuevo valor de s = ",s;
Imprimir "Nuevo valor de t = ",t;
```

Guarde el programa con el nombre **Ejprg08.txt** y ejecútelo.

Observando el código en Python, podemos apreciar que ambas instrucciones **for** y **while** presentan una indentación.

Ejemplo 18

En el siguiente programa se usa la función **Si** y **Repetir**, una después de otra.

```
# Uso de la sentencia Si y Repetir

Definir {

Var Icex, Atala, i: Entero;
Var s, t, u: Real;
}
s=1;
t=10;
u=0;
Leer "Numero 1: ", Icex;
Leer "Numero 2: ", Atala;
Si Icex>Atala entonces Imprimir ": ", Icex, "es mayor que ", Atala;
Si Icex < Atala entonces Imprimir ": ", Atala, "es mayor que ", Icex;
Imprimir "Uso de Repetir \nDesde s = ", s, "\nhasta t = ", t;
Repetir variando i desde s hasta t+1 {
u = u + i;
Imprimir "i = ", i, "u = ", u;
}

Imprimir ": ", Icex, Atala;
Imprimir "Suma = ", u;
```

Guarde el programa con el nombre EjPrg09.txt y luego ejecútelo.

En el caso de la instrucción Si, puede apreciar que no se ha usado las llaves para indentar sino por el contrario se ha usado entonces que permite colocar la única sentencia que se ejecutará cuando la comparación sea verdadera.

Otra novedad que apreciamos es el uso de control de línea “\n”.

Ejemplo 19

En este ejemplo vamos a usar una instrucción repetitiva dentro de la instrucción Si … Caso contrario … Del mismo modo veremos bajo qué condiciones se debe usar en este lenguaje que es para principiantes en programación y en Python.

El código es el siguiente:

```
# Programa ejemplo
#
Definir {
Var m, n, f: Entero;
Var s : Real;
#
Leer "Ingresa un valor para m: ", m:
Leer "Ingresa un valor para n: ", n:
#
f = 1;
suma = 0;
ndato = 0;
Si m < n {
Repetir variando i de 2 hasta m {
f = f*i;
}
p = f;
cad = "";
}
Caso contrario {
Repetir variando i desde 1 hasta m {
suma = suma + i;
ndato = ndato + 1;
cad = cad + str(suma)+"\n";
}
p = suma/ndato;
}
Imprimir "Sumas parciales:";
Imprimir cad;
Imprimir "El factorial o promedio de m numeros es = ", p;
```

Si m < n, se calcula el factorial de **m** usando, **Repetir**. En caso contrario, si m>= n, se suman los **m** primeros números y se calcula su promedio.

Los resultados en cada caso se guardan en **p**, lo que se imprime al final de los bucles.

Debemos apreciar también que en el segundo caso se guardan las sumas parciales en una variable de cadena cad. Finalmente se imprime los resultados en p y los que están almacenados en cad.

Guarde el programa como **EjPrg09a.txt** y luego ejecútelo.
Limitaciones del LCON:

Observe que no hemos usado ninguna instrucción del lenguaje al interior del bucle de **Repetir** pues si se usar **Leer** o **Imprimir** arrojaría errores de sintaxis pues el indentamiento no sería el indicado. Esto no ocurre con el Python que sí acepta cualquiera que fuera la forma de codificar sus instrucciones.

Ejemplo 20

En el siguiente programa, vamos a procesar n productos. En cada uno de ellos vamos a leer: Nombre del producto, cantidad de producto comprado y el descuento aplicado. El descuento puede ser de 0%, 10% ó 15%. Se aplicará el IGV.

Variables a usar:

n:	Número de productos
pName:	Nombre del producto
prUnit:	Precio unitario
Cantidad:	Cantidad
desc:	Descuento (0.0, 0.10, 0.15)
monto:	Importe
xTotal:	Importe total

El programa es el siguiente:

```
# Procesamiento de datos comerciales

Definir {
Var pName: Cadena;
Var prUnit, desc: Real;
Var n, Cantidad: Entero;
Var xTotal, monto: Real;
}
xTotal = 0;
Leer "Numero de productos: ", n;
Repetir variando i de 1 hasta n {
```

```
Leer "Nombre del producto: ", pName;
Leer "Precio unit: ", prUnit;
Leer "Cantidad: ", Cantidad;
Leer "Descuento: ", desc;
monto = Cantidad*prUnit;
monto = monto + monto*0.18-monto*desc;
xTotal = xTotal + monto;
Imprimir "Importe: ", monto;
}
Imprimir "Importe total: ", xTotal;
```

Guarde el programa como **EjPrg13.txt** y pase a ejecutarlo

CURIOSIDAD:

En el ejemplo anterior, se pudo haber independizado la lectura de los datos del proceso de cálculo. ¿Por qué no se hizo?

El manejo de arreglos, en este caso, el uso de los vectores resolvería esta pregunta.

NOVEDAD DEL LCON: EL USO DEL ARREGLO aX

El lenguaje LCON no maneja arreglos; es decir, vectores y matrices. Sin embargo, veremos más adelante, en la sección Aplicaciones estadísticas sí se maneja de forma indirecta.

Pero, para procesos especiales como en el caso del ejemplo anterior (**EjPrg13.txt**), este lenguaje usa el arreglo vectorial llamado aX (nombre reservado) donde podemos almacenar un conjunto de datos y desde la cual podremos visualizar los mismos en la pantalla.

En los siguientes dos ejemplos resolveremos la confusión que se tiene entre la lectura y cálculo de los datos.

ALMACENANDO DATOS EN aX

Para guardar datos en este arreglo se usa la instrucción TGuardar cuya sintaxis es:

TGuardar *cadenaDeCaracteres*

En donde ***cadenaDeCaracteres*** contiene los datos literales o numéricos formando una cadena de caracteres. Si ésta incluye valores numéricos, se debe tomarlos como como caracteres usando la función str().

Ejemplo 21

Veamos el siguiente ejemplo:

```
Definir {
Var pName: Cadena;
Var n: Entero;
Var cantidad: Entero;
Var prUnit, monto: Real;
}

Leer "Nro de productos: ",n;
Repetir variando i desde 1 hasta n {
Leer "Nombre: ", pName;
Leer "Cantidad: ", cantidad;
Leer "Pr. unit.: ", prUnit;

TGuardar pName+"---"+str(cantidad)+"---"+str(prUnit);
}
```

Puesto que el bloque de sentencias de la instrucción **Repetir** incluye a **TGuardar**, el arreglo **aX** almacena n elementos de tipo cadena.

Guarde este programa con el nombre **EjPrg13a.txt** y ejecútelo.

Como puede ver, la ejecución no genera ningún resultado. Y esto porque no se imprime ningún resultado.

Observe también el programa fuente generado en Python. Todas las sentencias del bloque están indentadas y no hay ninguna impresión.

Pregunta:

¿Cómo puedo imprimir aquello que se guardó con **TGuardar**?

Ejemplo 22

El siguiente programa responde a esta pregunta

```
#Procesando arreglos
```

```
Definir {
Var n: Entero;
Var aName: Cadena;
Var aCant: Entero;
Var aMonto, aprUnit: Real;
Var xTotalDeMonto: Real;
}

xTotalDeMonto = 0;
Leer "Nro de productos: ", n;

Repetir variando i desde 1 hasta n {
Leer "Nombre: ", aName;
Leer "Cantidad: ", aCant;
Leer "Precio: ", aprUnit;
aMonto = aCant*aprUnit+aCant*aprUnit*0.18;

TGuardar
"Prod:"+aName+","+"cantidad:"+str(aCant)+","+"pr.unit:"+str(aprUn
it)+","+"Monto:"+str(aMonto);
xTotalDeMonto = xTotalDeMonto + aMonto;

Imprimir "Monto: ", aMonto;
}

OImprimir aX;

Imprimir "Monto total: ", xTotalDeMonto;
```

Como en el programa anterior, se van a procesar **n** productos, en cada uno de ellos se lee el nombre, precio y cantidad. Se calcula el monto y se guarda todos esto con **TGuardar** además de imprimir el monto de la venta. Al terminar de procesar los n productos, se imprime lo almacenado en el arreglo **aX** usando la instrucción **OImprimir aX** y se termina imprimiendo el monto total de todas las ventas.

OImprimir aX es una frase reservada.

Guarde este programa con el nombre **EjPrg13b.txt** y ejecútelo.

Observe el programa traducido a Python.

La instrucción **OImprimir** de LCON ha generado la instrucción **for**:

```
for i in range(len(aX)):
        print(aX[i])
```

que es la forma de imprimir a cada uno de los elementos de un arreglo, línea por línea.

A continuación, vamos a terminar el capítulo analizando las diversas funciones que podemos emplear en el lenguaje LCON.

c. Funciones

El lenguaje **LCON** evalúa la mayoría de las funciones matemáticas y trigonométricas.

Limitación:

No permite el tratamiento de funciones definidas por el usuario.

La siguiente tabla nos muestra todas las funciones que pueden ser usadas en este lenguaje.

Función	Descripción
= entero(x)	Devuelve la parte entera de x
= raizc(x)	Devuelve la raíz cuadrada de x
= pi()	Devuelve el valor de π
= pot(x,n)	Devuelve la potencia de x a la n
= redondear(x,n)	Redondea a x con n decimales
= exp(x)	Devuelve e elevado a la x
= log10(x)	Devuelve el logaritmo decimal dex
= ln(x)	Devuelve el logaritmo neperiano de x
= aleat()	Obtiene un número aleatorio entre 0 y 1
= sen(x)	Devuelve el seno de x (x en grados)
= cos(x)	Devuelve el coseno de x (x en grados)

A continuación, veremos algunos programas en **LCON** que permite el uso de estas funciones.

Ejemplo 23

En el siguiente programa haremos uso de las funciones entero(), exp(), pi(), seno() y redondear.

```
# Veamos este programa
Definir {
Var a: Entero;
Var b, r: Real;
Var c, d: Real;

Leer "Ingresa un numero menor que 5: ", a;
Leer "Ingresa otro con decimales: ", b;

xb = entero(b);
c = exp(-a);
d = pi();
angulo = 30;
e = sen(angulo);
r = redondear(e,2);
Imprimir "Valor leido: ", b, " Su parte entera: ", xb;
Imprimir "Valor leido: ", a, " exp(-a)  = ", c;
Imprimir "Valor de pi: ",d, "\nSeno de 30 grados = ", e;
Imprimir "Valor del seno(30) redondeado con 2 decimales: ", r;
```

Guarde este programa con el nombre **EjPrg14.txt** y ejecútelo.

Recordemos que las funciones trigonométricas se calculan para valores en grados convertidos en radianes; sin embargo, LCON recibe grados e internamente se convierte a radianes.

Observe este detalle en el programa fuente generado en Python:

```
e = np.sin(angulo*np.pi/180)
```

np es una librería que se ha importado al inicio. La expresión pi/180 permite convertir a angulo en radianes.

Ejemplo 24

En el siguiente programa haremos uso de otras funciones matemáticas, así como las logarítmicas.

```
# Funciones matemáticas
Definir {
Var a: Entero;
Var b, r: Real;
Var c, d: Real;
```

```
Leer "Ingresa un numero mayor que 100: ", a;
Leer "Ingresa otro con decimales: ", b;

c = entero(b);
d = raizc(a);
e = raizc(d);
f = exp(-e);
g = ln(d);
xlog10 = log10(b);

Imprimir "Dado ", b, " su entero: ", c;
Imprimir "Raiz cuadrada de ", a, " es ", d;
Imprimir "Raiz cuarta de ", a, " es ", e;
Imprimir "exp(-",e, ") es ",f;
Imprimir "Logaritmo decimal de ", b, " es ", xlog10;
Imprimir "Logarimo neperiano de ", d, " es ", g;
```

Guarde este programa con el nombre **EjPrg14a.txt** y ejecútelo.

Sugerimos observar detenidamente la conversión de estas sentencias en Python.

Ejemplo 25

En el siguiente ejemplo seguiremos presentando el uso de funciones trigonométricas, pero bajo el contexto de la función **s**i.

```
Definir {
Var a,   b: Entero;
Var c, d: Real;
Var x, y, r, g, h: Real;

Leer "a = ", a;
Leer "b = ", b;
c = b;
d = a;
x = -6.28;
y = 6.28;
r = pi();
Imprimir "pi = ", r;
g = x/2*r;
h = y/2*r;
Si a < b {
c = 1/raizc(2*pi());
```

```
c = redondear(c,6);
d = sen(g);
d = d*sen(g);
e = cos(h);
e = e*cos(h);
d = d + e;
}
Imprimir "c = ", c;
Imprimir "d = ", d;
```

Guarde este programa con el nombre **EjPrg14b.txt** y ejecútelo.

Observe el valor de la variable d; es la suma de $sen^2(x)$ y $cos^2(x)$ que, como sabemos es 1.0.

Observamos también que, si el valor leído para **a** es mayor o igual que **b**, el programa arroja error.

Este programa se aprecia mejor en su traducción a Python, como lo puede apreciar la emisión del mismo, líneas arriba.

Ejemplo 26

El siguiente programa es similar al anterior, con la diferencia que el valor del cual se obtiene el seno y coseno resulta de una operación un tanto caprichosa y, cuando **a** >= **b**, ya no arroja error.

```
Definir {
Var a,   b: Entero;
Var c, d: Real;
Var x, y: Real;

Leer "a = ", a;
Leer "b = ", b;
x = -6.28;
y = 6.28;
c = log10(a);
d = ln(b);
r = pi();
h = x/2*r;
g = y/2*r;

h = sen(h);
g = cos(g);
Si a < b {
```

```
c = 1/raizc(2*pi());
c = redondear(c,6);
d = pot(h,2);
e = g**2;
d = d+e;
}
Caso contrario {
Imprimir "Por el caso contrario";
c = sen(b);
d = log10(a);
}
Imprimir "c = ",c;
Imprimir "d = ",d;
```

Guarde el programa con el nombre **EjPrg14c.txt** y proceda a ejecutarlo.

Ejecute ingresando valor para **a** menor que **b** y vuelva a ejecutarlo invirtiendo los valores de **a** y **b**.

Ejemplo 27

En el siguiente programa contemplaremos el uso de la instrucción Si { … } Caso contrario { … }

El programa es el siguiente:

```
Definir {
Var a,   b: Entero;
Var c, d: Real;
Var x, y: Real;

Leer "a = ", a;
Leer "b = ", b;
x = -6.28;
y = 6.28;
Si a < b {
c = 1/raizc(2*pi());
c = redondear(c,6);
d = sen(x/2);
d = d*sen(x/2);
e = cos(x/2);
e = e*cos(x/2);

d = d + e;
```

```
}
Caso contrario {
c = sen(x);
d = cos(y);
}
Imprimir "c = ", c;
Imprimir "d = ", d;
```

Guarde el programa con el nombre **EjPrg14d.txt** y pase a ejecutarlo.

Cuando **a** es mayor o igual que **b**, se calcula el seno y coseno de un valor positivo uno y negativo el otro.

Obsérvese, en la codificación en Python, líneas arriba, la elegancia de la programación estructurada que usa el Python al usar la forma indentada en el uso de la instrucción **if** … **else** …

Ejemplo 28

En el siguiente programa usaremos algunas funciones implicadas entre las instrucciones **Si** , **Repetir** y **Mientras**.

A continuación, el programa fuente:

```
# Programa usando Mientras
Definir {
Var a, b, n: Entero;
Var c, g: Real;
Var i: Entero;
Var s, t, u: Real;
}
s=0;
Leer "a = ", a;
Leer "b = ", b;
Leer "Grados: ", g;
s = sen(g);
t = cos(g);
Imprimir "Seno de ", g," = ", s;
Imprimir "Valor del coseno de g = ", t;
s = s*sen(g);
t = t*cos(g);
Imprimir "sen²60 = " s,", cos²60 = ", t;
s = 0;
t=0;
u = 0;
```

```
Repetir variando x de a hasta b {
s = s+raizc(b)*x;
t = t + aleat()*x;
u = u + cos(a*x);
}
Imprimir "Valor de s = ", s;
Imprimir "Valor de t = ", t;

Si s>t entonces Imprimir s," es mayor que ", t:
i=0;
s=0;
t = 0;
Mientras i<=10 {
s=s+a*i;
t = t + b*i/10;
i = i+1;
}
Imprimir "s = ", s;
Imprimir "t = ", t;
Imprimir "u = ", u;
```

Guarde el programa con el nombre **EjPrg15.txt** y ejecútelo.

Observe también, líneas arriba, la elegancia en cada una de las instrucciones traducidas al lenguaje Python.

Ejemplo 29

Terminaremos los ejemplos en el uso de las diversas funciones con un gráfico de tipo texto de la función seno construido con una programación muy simple.

Veamos el programa:

```
Definir {
Var a,    b: Real;
Var x: Real;
Var n: Entero;
}
a = -628;
b = 628;

#a= a/10;
#b = b/10;
c = int(a);
```

```
d = int(b);
Repetir variando i de c hasta d incr 10 {
x = 40+40*sen(i);
n = entero(x);
Imprimir "."*n;
}
```

En el ejemplo, a y b son valores en el eje X. Si estuviera dividido entre 100, representarían los valores extremos en una distribución Normal N(0, 1).

En el caso de usar valores de la función seno, y estar imprimiendo hacia abajo, **x = 40 + 40*sen(i)** permite obtener valores enteros en n, desde 0 hasta 80, valores que se están desplazando imprimiendo un punto ".".

Grabe el programa con el nombre **EjPrg16.txt** y ejecútelo.

d. Gráficos

La construcción de gráficos en LCON es elemental. Es suficiente una sentencia para trazar un determinado gráfico y quizás dos, si se desea incluir un texto o valor en el eje X.

Las demás propiedades adicionales de un gráfico se podrán completar programándolo en el lenguaje Python; el LCON ofrece sólo la parte inicial de Python.

a) Gráfico de línea

Un gráfico de línea permite analizar la tendencia de los datos; es decir, si éstos tienden a aumentar, disminuir o tienen una tendencia tipo serrucho.

Para construir un gráfico de línea es necesaria sólo una línea:

GLi[v1, v2, …, vn];

El siguiente programa nos muestra un gráfico de línea para las ventas de una bodega durante una semana.

Ejemplo 30

Mostramos el siguiente programa:

```
# Gráfico de línea
```

GLi[120,80,50,150,90,100,80];

Como puede apreciar, la palabra clave y reservada es GLi
Los datos se muestran en una lista (encerrados entre corchetes).

Guarde este ejemplo con el nombre **EjPrg17.txt** y ejecútelo

La numeración en el eje X, se muestra por defecto.

Ejemplo 31

El siguiente programa:

```
# Gráfico de línea
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GLi[120,80,50,150,90,100,80];
```

Se puede apreciar que, al ejemplo anterior, hemos incluido la sentencia que usa a X como palabra reservada, para insertar los valores que se van a incluir en el eje X. Estos valores pueden ser números, como en el ejemplo anterior, para el cual no se requiere de X, o como en este caso, valores tipo cadena.

Guarde este programa con el nombre **EjPrg17a.txt** y ejecútelo.

b) Gráfico de barras

Como en el caso anterior, construir un gráfico de barras en LCON es elemental.

Es suficiente una o dos sentencias de programación para visualizar un gráfico de barras.

Ejemplo 32

Dado el siguiente programa:

```
# Gráfico de barras
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GBa[120,80,50,150,90,100,80];
```

Hemos usado los mismos datos.

El título es insertado por defecto. Es un título estándar, para todos los gráficos de barras. La lista de colores contiene 4 elementos. La lista que se usa es:

colores = ["red", "green", "blue", "cyan"]

Si los datos a graficar tienen más de 4 valores, los colores se repiten.

Guarde este programa con el nombre **EjPrg17b.txt** y ejecútelo.

Ejemplo 33

En el siguiente programa vamos a trazar un gráfico de línea y otro de barras, con los mismos valores en el eje X.

```
# Gráfico de línea y barras
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GLi[120,80,50,150,90,100,80];
GBa[120,80,50,150,90,100,80];
```

Observe que las etiquetas usadas para el gráfico de líneas no es el mismo que para el gráfico de barras.

Sin embargo, si antepone la misma definición de X a la sentencia que grafica las barras podrá obtener los mismos valores en el eje X.

Guarde este programa con el nombre **EjPrg17c.txt** y ejecútelo.

Ejemplo 34

En el siguiente ejemplo, insertamos su propio eje para la segunda gráfica. De esta forma, el eje X ya no es definida por default.

```
# Gráfico de línea
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GLi[120,80,50,150,90,100,80];
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GBa[120,80,50,150,90,100,80];
```

Como lo dijimos en el ejemplo 32, los colores se repiten.

Guarde el programa con el nombre EjPrg17d.txt y ejecútelo.

Ejemplo 35

Ahora vamos a trazar un nuevo gráfico de barras pero que antes de ello se realiza algunas operaciones.

```
# Gráfico de línea

angulo = 30;
u = sen(angulo);
z = angulo*2;
y = cos(angulo*2);
Imprimir "seno de ", angulo," = ",u;
Imprimir "coseno de ", z," = ",y;
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GBa[120,80,50,150,90,100,80];
```

Grabe este programa con el nombre EjPrg17e.txt y ejecútelo.

c) Gráfico de torta

Pasemos ahora a construir los gráficos de torta o pie.

Ejemplo 36

Dado el siguiente ejemplo de programa:

```
# Gráfico de torta
GPi[80,50,20,90,150,120,80];
```

Podemos apreciar que sólo hemos cambiado la forma de invocar el procedimiento que permite construir el gráfico de línea dado en el ejemplo 30.

Como se puede ver, GPi es el procedimiento que invoca a la función que construye el gráfico de pie o torta.

El eje de categorías es la simple enumeración de las mismas como en ocurre en los gráficos anteriores.

Guarde este programa con el nombre **EjPrg18.txt** y ejecútelo

Ejemplo 37

En este caso estamos insertando los datos para definir el eje de categorías.

```
# Gráfico de torta
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GPi[80,50,20,90,150,120,80];
```

Guarde el programa con el nombre **EjPrg18a.txt** y ejecútelo.

Ejemplo 38

El siguiente programa permite construir los tres tipos de gráficos estadísticos vistos hasta ahora.

```
# Gráfico de torta
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GPi[80,50,20,90,150,120,80];
X[Ene, Feb, Mar, Abr, May, Jun, Jul, Ago, Set, Oct, Nov, Dic];
GLi[80,50,20,90,150,120,100,120,150,180,200,160];
X[Lun, Mar, Mie, Jue, Vie, Sab, Dom];
GBa[80,50,20,90,150,120,80];
```

Como se puede apreciar, los datos de las categorías lo hemos repetido sólo para mostrar que cada gráfico tiene sus propias categorías excepto que en el caso del gráfico de líneas los datos y categorías tiene una longitud de 12.

Grabe este programa con el nombre **EjProg18b.txt** y ejecútelo.

d) Gráfico de seno y coseno

Para graficar la sinusoide es suficiente ingresar los valores en el eje X.

La gráfica recorre el eje Y desde -1 hasta 1.

En el ejemplo siguiente, en el eje X, la gráfica va de -4 hasta 4

```
# Grafica del seno
GSe[-8,8];
```

Si desea guárdelo como **EjPrg19.txt** y ejecútelo.

En el siguiente ejemplo se muestra la gráfica de la función coseno desde -8 hasta 12

```
# Grafica del coseno

GCo[-8,12];
```

Si desea guárdelo como **EjPrg19a.txt** y ejecútelo

En el siguiente ejemplo estamos graficando las dos funciones trigonométricas

```
# Grafico de seno y coseno
GSe[-8,8];
GCo[-4,12];
```

Si desea guárdelo como **EjPrg19b.txt** y ejecútelo.

Observe que, siendo el rango diferente, el número de curvas varía.

Ejemplo 39

En el siguiente ejemplo se calcula valores de seno y coseno para valores de **a** hasta **b**, que son leídos previamente.

Cuando termina el primer proceso (iteraciones incrementando de increm cada vez, se procede a construir el gráfico del seno; vuelve a pedir valores de **a** y **b**, grafica ahora la función coseno y vuelve a pedir valores de **a** y **b**.

El programa es el siguiente:

```
Definir {
Var a,    b: Entero;
}
Imprimir "Desde a hasta b";
Leer "a = ", a;
Leer "b = ", b;
increm = 3;
Imprimir "Se incrementara en 3, cada iteracion";
Imprimir "Despues del primer grafico,\ningrese a y b nuevamente";
```

```
Repetir variando i de a hasta b inc increm {
y = sen(i);
x = 80+y*40;
Imprimir "i = ", i, " x = ", x;
}
GSe[-8,8];
GCo[-4,12];
```

Grabe este programa con el nombre **EjPrg19c**.txt y ejecútelo

Ejemplo 40

En el siguiente programa se obtiene dos valores enteros inicial **a** y final **b** de forma aleatoria. El valor aleatorio generado para a se ha multiplicada por 10 (el valor debe estar entre 0 y 9). El valor para b se ha multiplicado por 3000 por lo que el b será un valor entre 0 hasta 299:

```
Definir {
Var a,    b: Entero;
}
Imprimir "Se procesará desde a hasta b";
a = entero(10*aleat());
b = entero(aleat()*3000);
Imprimir "Inicial = ", a;
Imprimir "Final = ", b;
Repetir variando i de a hasta b incr 10 {
x = 40+40*sen(i);
n = entero(x);
Imprimir "."*n;
y = 40+40*cos(i);
m = entero(y);
Imprimir "o"*m;
}

GSe[-8,8];
GCo[-4,12];
```

En cada iteración se generan valores de **x** e **y** cuyos valores van de –a hasta 1, los que sumados a 40, generan valores entre 0 y 80.

Finalmente se imprime un “*” en el caso del seno y “o” en el caso de coseno, lo que genera el gráfico de texto como se puede apreciar al ejecutar el programa.

Al final del programa se construye el gráfico del seno y del coseno.

Estos gráficos increíblemente se muestran, pero para cada generación de a y b, mostrando tres tablas de longitud aleatoria.

Guarde este programa con el nombre **EjPrg19d.txt** y ejecútelo.

e) Gráfico de funciones polinomiales

En adición a los gráficos hasta ahora vistos, presentaremos la construcción de gráficos de funciones polinómicas de grado n.

Estas funciones van a ser de dos tipos:

1. Gráficos polinomiales demostrativo

   En este caso el programa fuente es un demo; es decir, se construye la gráfica de la función polinómica, de la derivada de la función y de la integral de la misma, obteniendo tres funciones que se pasan a graficar usando algunas de las propiedades del ploteo y que la librería matplorlib nos permite.

   En efecto,
   Si se tiene lista = [150, 60, -3, -12
   Entonces p(x) = 150 + 60*x – 3x**2 – 5x**3
   Derivada de p(x) = 60 – 6x – 36x**2
   Integral de p(x) = 150x + 30x**2-x**3-3x**4

   Con esta idea pasamos a presentar el programa fuente en LCON que nos permita la construcción de un gráfico para un polinomio p(x).

   Dado p(x) = a + bx + cx**2 + dx**3 + … + rx**n
   Podemos pensar en su derivada y en la integral de la misma.

   La función que vamos a usar nos permitirá graficar a p(x), la derivada de p(x) y a la integral de p(x)

   Ejemplo 41

El siguiente ejemplo muestra el programa fuente:

```
# Gráficos de
#La función polinómica p(x)
# La derivada de p(x)
# La integral de p(x)
#
GPo
#
```

Como puede apreciar, es suficiente la sentencia GPo que invoca la gráfica de estos tres polinomios.

Si desea guarde el ejemplo con el nombre **EjPrg20.txt** y ejecútelo.

NOTA

Luego de ejecutarlo, la función emitirá unos mensajes y pedirá que se ingrese:

- El grado del polinomio
- Todos los coeficientes del polinomio

2. Gráficos polinomiales dada la función polinomial

   En esta opción de trazado de gráfico polinomial, tendremos un programa fuente más interactivo.

   El programa es el siguiente:

```
# Gráfico de un polinomio p(x) de grado n
#
GCu[-2, 2];
#
```

   Si y = p(x), entonces el gráfico corresponderá a un polinomio de una variable de grado n.

   Al ejecutar el programa, éste mostrará un pequeño menú donde se debe seleccionar si se desea graficar un polinomio bi o tridimensional. Luego se debe ingresar el polinomio p(x) correspondiente.

   Guarde este ejemplo con el nombre **EjPrg20a**.txt y ejecútelo.

f) Evaluación de polinomios

Ahora vamos a ver cómo se puede evaluar polinomios de grado n.

Ejemplo 42

Como un nuevo ejemplo veamos cómo se puede evaluar un polinomio p(x) de grado n.

El siguiente es el programa fuente:

```
#Evaluación de una expresión p(x) o p(x, y)
GEv(5);
GEv(2,3);
```

En la primera instrucción de este ejemplo se desea evaluar una función polinómica como p(x) = 5*x**3-2*x+2.

En la segunda instrucción se pretende evaluar una función por ejemplo p(x) = 3*x**2*y**3+3*x*y-x**2-y**2+10

Estas funciones se ingresarán cuando el **IceCompilador** se lo pida.

Guarde este programa con el nombre **EjPrg20b.txt** y ejecútelo.

g) Gráficos 3D (de superficie)

Esta es la última sección de gráficos, en la cual presentaremos gráficos de superficie o comúnmente llamados gráficos 3D.

LCON es un lenguaje muy simple pero en algunas de sus instancias invade terrenos del Python a profundidad como ya lo hemos visto en múltiples ocasiones.

Una de esas oportunidades es ésta.

La construcción de gráficos 3D perteneces a estudios avanzados del Python. Sin embargo, con el lenguaje LCON, lo hacemos de manera

muy elemental y simple. Esto lo hemos visto con gráficos estadísticos y polinomiales.

Ejemplo 43

En este programa presentamos las instrucciones necesarias para invocar la construcción de gráficos 3D:

```
# Gráfico 3D
Imprimir "Gráficos tridimensionales","\n O de superficie";
G3d;
```

Es simple, sólo requiere invocarla con **G3d** y listo.

Al ejecutar este programa veremos la construcción de un gráfico de superficie cuya esencia es la función x**2 + y**2 y cuya curva de nivel lo proporciona la función seno y coseno.

Usando el ratón o los dedos (en una pantalla táctil) mueva la gráfica para cambiar su forma visual en los ejes coordenados.

Luego de esta gráfica se muestra otra que en matemática se le conoce como la silla de montar.

Podemos modificar los datos que modifican su contexto, pero eso será para una nueva versión del lenguaje LCON.

Guarde el programa con el nombre **EjPrg21.txt** y ejecútelo.

NOTA:

El lenguaje LCON, bajo la perspectiva de las mínimas líneas de instrucción que se requiere para crear diversos tipos de gráficos, podemos decir que **ES UN LENGUAJE DE MACROS**, como lo era el Assembler de la IBM/1130 por ejemplo allá por los años 60 a 80 del siglo pasado.

En dicho lenguaje, una sola macro instrucción, por ejemplo, para leer una tarjeta de 80 columnas, reemplazaba a decenas de instrucciones del Assembler nativo que debería orientar su dispositivo de entrada a una lectora y no al monitor.

Ejemplo 44

Como un segundo ejemplo de programa fuente que permite invocar la construcción de gráficos de superficie aleatoria, tenemos el siguiente:

```
# Dos gráficos de superficie
G3a
```

A diferencia del ejemplo anterior, en este caso se invoca a un módulo que permite la visualización de gráficos de superficie seleccionados de manera aleatoria.

Cada vez que ejecute este programa verá usted que el gráfico de superficie es otro.

Grabe este programa con el nombre **EjPrg21a.txt** y ejecútelo.

En este caso se genera un número aleatorio r, entero entre 0 y 20.

Si r <= 5 entonces la superficie a graficar es: Z = 2*Y*X + X**2 + Y**2

Si r > 5 y r <= 10, la superficie elegida es Z = 36-4*X**2-9*Y**2

Si r > 10 y r <= 15, la superficie elegida es Z = np.sin(X)+np.cos(Y)

En caso contrario la superficie seleccionada es Z = x**2 - y**2.

CAPITULO V

V. APLICACIONES ESTADÍSTICAS

En este capítulo desarrollaremos la potencia de LCON en el tratamiento de los datos y cálculos estadísticos.

La lectura, por lo general se hará desde un archivo de texto, como los programas fuentes que estamos usando.

Los datos se grabarán usando un editor de textos y sea por fila o columna estarán separados por “,” o por un tabulador “\b” y, si están grabados hacia abajo, se tomará en cuenta el fin de línea cuyo código es “\n”.

a. Estadística descriptiva

En esta sección nos dedicaremos a desarrollar todas las herramientas que tiene la Estadística Descriptiva para una variable.

Calcularemos las medidas de tendencia central como la media, la mediana y los percentiles

Invocaremos también la tabla de frecuencias

Del mismo modo calcularemos la varianza, el coeficiente de variación y los coeficientes de simetría.

El IceCompilador está equipado de una serie de funciones estadísticas que las invocaremos desde los programas fuentes de manera elemental, como hasta ahora venimos haciendo con el lenguaje LCON.

El número de variables que tomaremos en cuenta para las herramientas de la Estadística Descriptiva será 1.

Ejemplo 45

En este ejemplo vamos a codificar las instrucciones del programa en vivo; es decir, la iremos digitando línea por línea y la iremos comentando.

En cada nueva modificación o añadido, grabaremos el programa y pasaremos a ejecutarlo

Empecemos:

Abra el editor.
Digite:

Aplicaciones estadísticas
Esto es un comentario, no va a ser procesado
En la siguiente línea digitamos:

Estad;

Grabe este programa de dos líneas con el nombre de **EjPrg22.txt** y pase a ejecutarlo.

OBSERVACIÓN IMPORTANTE

No está demás volver lo que dijimos en el numeral b de la página 10 que para ejecutar nuestros programas fuentes, debemos tener abierto el IceCompilador01 y hacer clic en <RUN> - <Run module> lo que nos permitirá ingresar la ruta y nombre del programa fuente. En este ejemplo y en mi caso: **d:/pyCcon/EjPrg22.txt**

Esta sentencia sólo invoca a un módulo del compilador que se encarga de activar todas las funciones disponibles y que están listas para ser usadas.

En la pantalla se nos pide que digitemos el número de variables. Como estamos procesando herramientas de la Estadística Descriptiva, se trata de una variable, como tal digitamos 1

Ahora se muestra un pequeño **menú** con dos opciones:

Ingreso de los datos:
1. Desde el teclado
2. Desde un archivo

Digite su opción:

Digitamos 1 pues los datos los vamos a ingresar desde el teclado.

En seguida nos pide que digitemos el número de datos. Si digitamos 12, se nos pedirá que ingresemos 12 datos numéricos enteros y/o reales según nuestro interés.

Al final termina el ingreso de datos y termina también de ejecutarse el programa.

Como puede ver, no se emite ningún resultado. Claro, si no le hemos pedido nada.

Ahora modifique el programa de forma que contenga las siguientes líneas:

```
# Aplicaciones estadísticas
Estad
Imprimir “Los datos a ser procesados: “, X;
```

Vuelva a grabar el programa con el mismo nombre y pase a ejecutarlo.

Ahora tendrá la lista que contiene los datos ingresados.

Agregaremos una línea más donde le pediremos que nos imprima la longitud de la lista; es decir, el número de datos. El programa debe quedar como sigue:

```
# Aplicaciones estadísticas
Estad;
Imprimir X;
n = len(X);
Imprimir "Nro de datos a procesar: ", n;
```

La función **len()** permite obtener el número de elementos o tamaño de la lista X.

Grabe el programa y con el mismo nombre y vuelva a ejecutarlo

Ejemplo 46

Continuando con el programa anterior, ahora vamos a usar lo que hemos aprendido para calcular la media, la varianza, la desviación estándar y el coeficiente de variación.

Para calcular estas herramientas estadísticas es necesario tener la suma de los datos y la suma de sus cuadrados como se muestra en la siguiente imagen

$$mx = \frac{\sum X_i}{n} \qquad vx = \frac{\sum\left(X_i - \overline{X}^2\right)^2}{(n-1)} = \frac{\sum X^2 - n\overline{X}^2}{(n-1)}$$

Usaremos a s y s2 inicializadas con

```
s = 0;
s2 = 0;
```

para obtener las dos sumatorias usando la instrucción **Repetir n veces**

```
s = s + X[i];
s2 = s2 + pot(X[i],2);
```

Luego de calcular las sumas procedemos a calcular las herramientas

```
mx = s/n;
vx = (s2-n*mx**2)/(n-1);
desvx = raizc(vx);
cvarx =desvx/mx;
```

Ahora pondremos las impresiones de estos indicadores:

```
Imprimir "Media de X: ", mx;
Imprimir "Varianza de X: ", vx;
Imprimir "Desv- estandar de X: ", desvx;
Imprimir "Coef de variacion de X: ", cvarx;
```

El programa queda de la siguiente forma:

```
# Aplicaciones estadísticas
Estad;
Imprimir X;
# La función len() devuelve la longitud de la lista
n = len(X);
Imprimir "Nro de datos a procesar: ", n;
s = 0;
s2 = 0;
Repetir n veces {
s = s + X[i];
s2 = s2 + pot(X[i],2);
```

```
}
mx = s/n;
vx = (s2-n*mx**2)/(n-1);
desvx = raizc(vx);
cvarx =desvx/mx;
Imprimir "Media de X: ", mx;
Imprimir "Varianza de X: ", vx;
Imprimir "Desv- estandar de X: ", desvx;
Imprimir "Coef de variacion de X: ", cvarx;
```

Guarde el programa con el nombre **EjPrg22a.txt** y ejecútelo.

Al ejecutarlo, como en el programa anterior

Digitaremos el número de variables: 1
En el menú digitamos 1 pues los datos los vamos los datos desde el teclado.

A continuación, ingresamos el número de datos y procedemos a ingresar todos los n datos.

Luego de esto obtendremos los resultados.

Ejemplo 47

Usando el editor vamos a grabar los siguientes datos:

350,180,165,195,180,210,240,180,210,165,150,285,127,120,135,195,240,270,300,330

Grabarlo con el nombre **d01a.txt** en la unidad y carpeta que esté usando.
En mi caso lo tengo en d:/pyCcon

Ejecutemos el programa anterior

Digitemos 1 cuando pida número de variables.
En el menú digitamos 2 para seleccionar la opción que permite leer los datos desde un archivo.

Cuando pida nombre del archivo, en mi caso digito: **d:/pyCcon/d01a.txt**

De inmediato veremos la solución correspondiente.

Ejemplo 48

A continuación, vamos analizar el siguiente programa fuente en LCON que, como se puede apreciar, permitirá digitar los datos desde el teclado o se leerán desde un archivo.

He aquí el programa fuente:

```
# Estadísticas de una variable
Definir {
Var X: Vector;
Var xcad: cadena;
}
Estad;
mx = media(X);
vx = var(X);
mediana = mediana(X);
q = cuartiles(X);
Imprimir "Media: ", mx;
Imprimir "Varianza: ", vx;
Imprimir "Mediana: ", mediana;
Imprimir "Primer cuartil: ", q[0];
Imprimir "Segundo cuartil: ", q[1];
Imprimir "Tercer cuartil: ", q[2];
```

Comentarios al respecto:

Observe que, ante todo, no estamos realizando el cálculo.
Para la media se invoca una función llamada **media()**
Para la varianza se invoca la función var()
Para la mediana se invoca la función **mediana()**
Para los tres cuartiles se invoca la función **cuartiles()**

Puede apreciar que la función **cuartiles()** devuelve una lista de tres valores que lo recibe q que se convierte en una lista de tres elementos.

Guarde este programa como **EjPrg23.txt** y ejecútelo

Número de variables: 1
Opción: 2 (Leer datos desde archivo)
Nombre del archivo de datos: **d:/pyCcon/d01a.txt**

La gran diferencia es que, en este caso, no se realiza ningún cálculo, Sólo se invoca diversas herramientas (funciones) para obtener el estadístico.

Volvamos a ejecutar el programa y, usando 1 variable y la opción que permite leer desde un archivo, ingrese como nombre de archivo: **d:/pyCcon/ingreso.txt**

Los archivos de datos: **d01.txt**, **d02.txt** y **d03.txt** son archivos que contienen valores para dos variables.

Archivo **d01.tx**t:

```
150     26
180     32
165     28
….
```

Archivo **d02.txt**:

```
150     180     165     195     180     210     240     180     210     165
   150  285     127.5   120     135     195     240     270     300     330
26 32   28      34      35      32      42      28      36      26      28
   52   24      28      32      36      44      52      64      72
```

Es decir, son datos para realizar cálculos estadísticos para dos variables X e Y. Sin embargo, el programa sólo contempla cálculos para la primera variable (X).

Esto significa que devolverá resultados para esa variable.

En cuanto a los archivos de datos el primero tiene dos columnas separados por un tabulador (\t)

En el segundo, están separadas por un tabulador, pero están por fila

Y en el tercer archivo, están por fila, pero los elementos de cada variable están separados por una coma.

Vuelva a ejecutar el programa **EjPrg23.txt** y digite 2 cuando pida el número de variables. En este caso, verá que no pasa por el menú de opciones.

Cuando pida el nombre de archivo de datos, digite: **d:/pyCcon/d01.txt**.

Haga lo mismo con el otro archivo, volviendo a ejecutarlo desde RUN – RUN module del IceCompilador01.

En cada caso obtendrá resultados para la primera variable.

El siguiente programa es una modificación del **EjPrg23.txt**.

```
# Estadísticas de una variable
Definir {
Var Y: Vector;
}
Estad;
my = media(Y);
vy = var(Y);
medianaY = mediana(Y);
qy = cuartiles(Y);
Imprimir "Media: ", my;
Imprimir "Varianza: ", vy;
Imprimir "Mediana: ", medianaY;
Imprimir "Primer cuartil: ", qy[0];
Imprimir "Segundo cuartil: ", qy[1];
Imprimir "Tercer cuartil: ", qy[2];
```

Es el mismo anterior solo que ahora se pide estadísticas de la variable Y.

Con el archivo d01.txt arroja un error en el cálculo de un estadístico no pedido.

En los archivos d02.txt no hay errores.

Ejemplo 49

Como último ejemplo de la Estadística Descriptiva veremos el siguiente programa que toma en cuenta a las dos variables cuando se trata de evaluar las dos variables.

El programa es el siguiente:

```
# Estadísticas de una variable
Definir {
Var X, Y: Vector;
}
Estad;
mX = media(X);
vX = var(X);
medianaX = mediana(X);
qX = cuartiles(X);
```

```
mY = media(Y);
vY = var(Y);
medianaY = mediana(Y);
qY = cuartiles(Y);
Imprimir "Media de X: ", mX, " Media de Y: ", mY;
Imprimir "Varianza de X: ", vX, " Varianza de Y: ", vY;
Imprimir "Mediana de X: ", medianaX, " Mediana de Y: ", medianaY;
Imprimir "1er cuartil de X: ", qX[0], " 1er cuartil de Y: ", qY[0];
Imprimir "2do cuartil de X: ", qX[1], " 2do cuartil de Y: ", qY[1];
Imprimir "3er cuartil: de X: ", qX[2], " 3er cuartil de Y: ", qY[2];
```

Guarde este programa con el nombre **EjPrg23b.txt**.

Al ejecutarlo debe digitar 2 cuando se le pida el número de variables.

Con el archivo de datos d01.txt arroja un error pero no impide obtener las estadísticas solicitadas.

Con los otros archivos **d02.txt** se obtiene los resultados solicitados.

b. Regresión lineal

El IceCompilador01 permite realizar un análisis de regresión lineal; es decir, dada la función

$Y = f(X) = \mathbf{a} + \mathbf{b}X + u$

Donde u es una variable aleatoria llamada variable de perturbación, la cual se supone tiene una distribución N(0, 1) se puede estimar los coeficientes de regresión **a** y **b**.

El lenguaje LCON permite codificar programas muy simples que permiten invocar las funciones estadísticas y también el procedimiento de regresión lineal llamada **regre()**.

NOTA

Siendo X la variable independiente y Y la dependiente, debemos tener cuidado de ingresar los datos.

En nuestro caso, la primera variable (y como se acostumbra) corresponderá a la variable dependiente; es decir, Y. Luego de ella se tendrá los datos para la variable X.

Y se toma de esta forma pues, en el caso multilineal, las variables explicativas (independientes) pueden ser X1, X2, …, etc.

Los datos deben ser digitados en forma vertical, Y, X1, X2, … separados por coma.

Una muestra:

El archivo de datos **ahorro.txt** contiene a Y: Ahorro, X1: Ingreso. Claramente podemos saber que la variable Y depende de X1.

```
3.2,20
3.4,21.1
3.6,22.4
3.2,21.2
2.85,15
3.1,18
2.85,18.8
3.05,15.7
2.7,14.4
2.75,15.5
3.1,17.2
3.15,19
2.95,17.2
2.75,16.8
2.95,18.5
```

Ejemplo 50

Antes de realizar un análisis de regresión propiamente dicha, vamos a evaluar las estadísticas de las variables X e Y mediante el siguiente programa:

```
# Estadísticas de dos variables
Definir {
Var X, Y: Vector;
Var xcad : cadena;
}
Estad();
Imprimir "Calculos estadisticos";
mX = media(X);
```

```
mY = media(Y);
vX = var(X);
vY = var(Y);
covXY = cov(X,Y);

Imprimir "Estadistico:";
Imprimir "============";
Imprimir "Media de X = ",mX;
Imprimir "Media de Y = ",mY;
Imprimir "Var(X) = ",vX;
Imprimir "Var(Y) = ",vY;

Imprimir "Cov(X,Y) = ";
Imprimir covXY;

Imprimir "=============";
```

Sólo queremos resaltar que la sentencia que permite activar el procedimiento para realizar un análisis de regresión será la sentencia **Estad()**.

Las otras líneas son similares a lo que ya hemos visto antes.

Guarde el programa con el nombre **Datos02.txt** y ejecútelo.

Tome nota de que ahora el número de variables que se deberá ingresar cuando lo pida es 2.

Cuando pida el nombre del archivo de datos se debe ingresar la ruta y el nombre del archivo: **d:/pyCcon/d02.txt** o cualquier otro que pudiera grabar usted en la forma del **d01.txt, d02.txt, d03.txt**, **ahorro.txt**, **ingreso.txt**, etc.

Luego del cual se verá los resultados.

La novedad en este caso ha sido el cálculo de la covarianza de X e Y (cov(X, Y) lo que nos permite saber si las variables X e Y presentan alguna relación de dependencia.

Ejemplo 51

En este ejemplo sí nos metemos de lleno al análisis de regresión lineal. Para ello, además de usar la sentencia Estad() que nos permite el acceso a las estadísticas descriptivas, también usamos la función regre() que nos

devuelve todo el bloque de cálculos implicado en un análisis de regresión lineal de dos variables.

He aquí el programa:

```
Definir {
Var fn, datos: Cadena;
Var nv: Entero;
}
Estad();
mediaX = media(X);
mediaY = media(Y);
varX = var(X);
varY = var(Y);
tempo = varX**0.5;
cvar = tempo/mediaX;
corXY = cor(X,Y);
Imprimir "Media de X = ",mediaX;
Imprimir "Varianza de X = ",varX;
Imprimir "Covarianza = ", cvar;
regresion = regre(Y, X);
Imprimir "Regresion lineal:";
Imprimir "=================";
Imprimir regresion;
Imprimir "=================";
Imprimir("ro(X,Y) = ",corXY;
Imprimir "Media de X = ",mediaX, "Varianza = ",varX;
```

Obsérvese que la función regre(Y, X) nos devuelve todo el análisis de estimación lineal completo.

En ella se incluye la ecuación estimada y también el ANOVA o la tabla del análisis de la varianza, así como el coeficiente de correlación y también el coeficiente de determinación indicándonos el primero el grado de relación que encontramos entre X e Y y qué porcentaje de explicación nos proporciona la variable explicada Y con la variable explicativa X.

Guarde el programa con el nombre Datos03.txt y ejecútelo.

Comol nombre del programa fuente yo digitaré **d:/pyccon/Datos03.txt**.

El número de variables será **2**

Y el archivo de datos será: **d:/pyCcon/d02.txt**

Ejemplo 52

¿Qué ocurre si sólo queremos realizar un análisis de regresión lineal y no los estadísticos descriptivos?

Veamos el siguiente ejemplo en el cual se invoca primero al procedimiento **Estad**(), para activar las herramientas estadísticas y luego se invoca a la función **regre()**.

El programa es el siguiente:

```
Definir {
Var fn, datos: Cadena;
Var nv: Entero;
}
Estad();
reg = regre(Y, X);
Imprimir reg;
```

Pudimos haber evitado la declaración de variables Definir {} que está demás.

Grabe el programa como Datos04.txt y ejecútelo

Cuando el IceCompilador01 pida el nombre del programa fuente, digitaremos d:/pyCcon/Datos04.txt

Cuando pida número de variables, digitamos 2

Cuando pida nombre del archivo de datos digitamos: d:/pyCcon/d02.txt

Se visualizará estrictamente todas las estadísticas de regresión.

Ejemplo 53

Para terminar esta sección, ejecutaremos el siguiente programa fuente en donde se pide todas las estadísticas descriptivas y de regresión.

```
# Ejemplo de uso de archivo de datos
Definir {
Var fname: Cadena;
Var X,Y : Vector;
Var datos: Cadena;
Var nv: Entero;
Var sX, sX2, sY, sY2, sXY: Real;
Var s: Real;
Var mX, mY,vX, vY, covXY, corXY: Real;
Var nv: Entero;
Var fn; Cadena;
}

Estad();
Imprimir("Cálculos:")
mX = media(X);
mY = media(Y);
vX = var(X);
vY = var(Y);
covXY = cov(X, Y);
corXY = cor(X,Y);
reg = regre(Y, X);
Imprimir "Estadístico:";
Imprimir "============";
Imprimir "Media de X = ",mX;
Imprimir "Media de Y = ",mY;
Imprimir "Var(X) = ",vX;
Imprimir "Var(Y) = ",vY;
Imprimir "=========";
Imprimir "Cov(X,Y) = ";
Imprimir covXY;
Imprimir "Regresión Lineal:";
Imprimir reg;


Imprimir "=========";
Imprimir "ro(X,Y) = ";
Imprimir corXY;
```

Obsérvese la abundante cantidad de variables que se declara al inicio. Como hemos visto, cuando una variable no va a ser usada o va a ser calculada, no es imprescindible la definición de ellas.

Grabe el programa con el nombre **Datos05.txt** y ejecútelo.

Cuando pida el nombre del programa fuente, digite: la ruta y nombre del archivo programa, yo digitaré:

d:/pyCcon/Datos05.txt

Cuando pida el nombre del archivo de datos, ingrese **ahorro.txt** (contiene como datos, cantidad ahorrada (Y) y el ingreso (X).

En mi caso digitaré:

d:/pyCcon/ahorro.txt

Ejemplo 54

Ahora haremos uso del programa fuente **Datos04.txt** y los datos **gastosTV.txt.** Los dos están en la carpeta **d:/pyCcon**.

Abra el IceCompilador01. Ejecútelo usando Run – Run model.

Cuando pida nombre del programa fuente, digite (en mi caso) **d:/pyCcon/Datos04.txt**

Cuando pida nombre de variables, digite **2**

Cuando pide nombre del archivo de datos digite (en mi caso) **d:/pyCcon/gastosTV.txt**

Como puede apreciar, se obtiene los resultados estimados en un análisis de regresión lineal simple de dos variables.

Se muestra los coeficientes de la regresión estimados, el coeficiente de determinación R^2 , el coeficiente de determinación corregido, el coeficiente de correlación, el error típico y fundamentalmente, la tabla del análisis de la varianza (ANOVA)

Luego de emitirse estos resultados, se completa la salida con los resultados que se pide en el programa **Datos04.txt**

c. Regresión multilineal

Una de las secciones más implementadas es ésta, el análisis de Regresión Lineal Múltiple.

Hagamos una presentación teórica así como la estimación de esta herramienta utilizando el Método de los Mínimos Cuadrados.

Dada la función

$Y = f(X_1, X_2, \ldots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + B_p X_p + u$

debemos obtener la función estimada $\hat{Y}$

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p + \varepsilon$$

El objetivo del método de los mínimos cuadrados consiste en obtener los $\hat{\beta}_i$para cada uno de los β_i, coeficientes de la regresión.

El método, así como todas herramientas principales del análisis de regresión múltiple, se encuentran implementado en el compilador IceCompilador01 a la cual tendremos acceso de forma muy elemental como cuando la hacíamos en otros casos y en le Regresión Lineal simple, desarrollada en la sección anterior.

Ejemplo 55

El programa fuente en LCON que usaremos para resolver problemas de esta sección será uno muy simple, de una sola sentencia:

```
# Regresión lineal múltiple
Estad()
```

Guarde este programa con el nombre de **Datos07.txt**.

1. Abra el IceCompilador01
2. Use <Run> - <Run module> para ejecutarlo
3. Cuando pida nombre del programa fuente digite (en mi caso): d:/pyCcon/Datos07.txt
4. Cuando pida el número de variables, digite 3
5. Cuando pida el nombre del archivo de datos digite (en mi caso) d:/pyCcon/distpc.txt

A continuación, verá dos ventanas de gráfico: una mostrando dos gráficas de dispersión en la misma ventana. Y una segunda, otra ventana dividida en 4 partes para mostrar el gráfico de dispersión de la variable dependiente Y, con cada una de las variables independientes: X1, X2.

En el panel del IDLE se tendrá una gran cantidad de resultados correspondientes al análisis de regresión lineal múltiple para terminar mostrando todas las estimaciones relativas a la tabla del análisis de varianza (Anova).

Ejemplo 56

Ahora obtendremos todas las estadísticas para 4 variables.

Use el mismo programa fuente.

Use los pasos1, 2 y 3 del ejemplo anterior.

Cuando pida el número de variables, digite 4

Cuando pida el nombre del archivo de datos, digite: d:/pyCcon/agrimil.txt

Ahora podrá visualizar los resultados cuando se trata de 4 variables del modelo lineal.

Nota:

No hacemos ninguna interpretación de los resultados por cuanto no estamos desarrollando el tema estadístico correspondiente.

CAPITULO VI

VI. EJEMPLOS Y EJERCICIOS

En este capítulo nos dedicaremos a desarrollar una serie de ejemplos que nos permitan crear los archivos de programa fuente as como los programas fuentes

Del mismo modo, plantearemos una serie de ejercicios para el usuario de ambos tipos de archivos de forma que faciliten el uso del lenguaje, permitan el uso del lenguaje LCON y lo fundamental: Permitan el conocimiento del lenguaje Python en forma básica.

Para grabar uno u otro archivo usaremos un editor de texto como por ejemplo el Bloc de notas de Windows. Hay muchos editores

El archivo se debe guardar con extensión txt.

a. Archivo de programa fuente

Escribir o codificar un programa fuente usando el lenguaje LCON pasa por digitar el código línea por línea usando un editor de texto.

Todos estos archivos deben tener extensión **txt**.

Toda línea que contiene al símbolo '**#**' se toma como comentario, los cuales no son tomados en cuenta para ser compilados por el compilador IceCompilador01.

Se puede usar tantas líneas comentario que ayude a comprender fácilmente lo que la o las sentencias del programa

Un archivo de programa fuente puede contener una línea de código o muchas líneas. Todas ellas terminan con el símbolo '**;**'

Un programa fuente tiene dos secciones:

La primera sección de definición de las variables según el tipo usando:

```
Definir {
Var varnombre1: tipo;
Ver varNombre2, varNombre3: tipo:
…
}
```

Las variables usadas en la sentencia Leer … son las que deben ser definida necesariamente. Es opcional para otras variables.

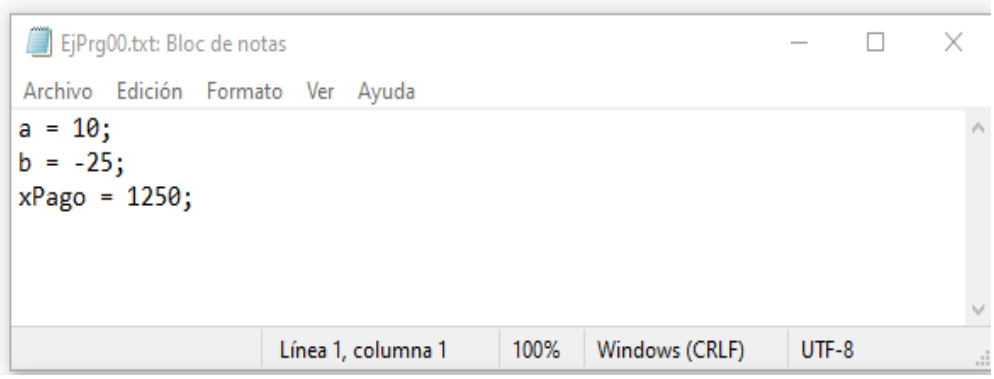
Todas las variables que son usadas como acumuladores o para resolver problemas de sumatorias, deben ser inicializadas por lo general en 0.

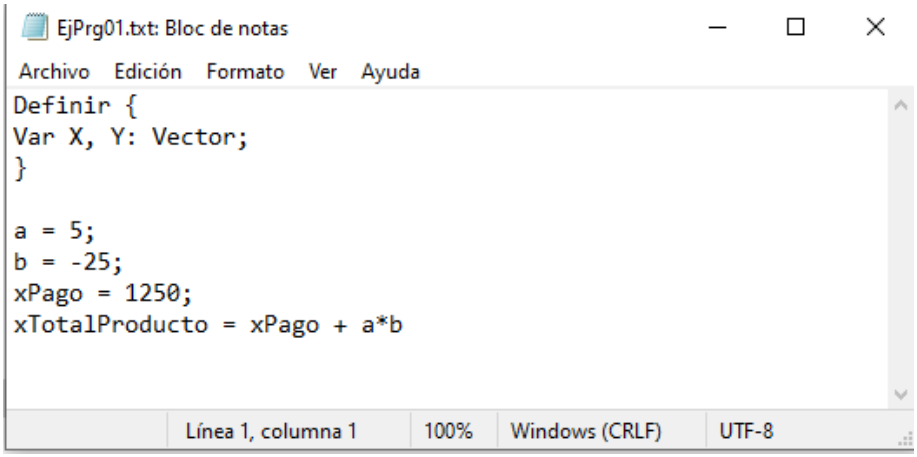
```
s = 0;
xPlan = 0;
cad = “”;
```

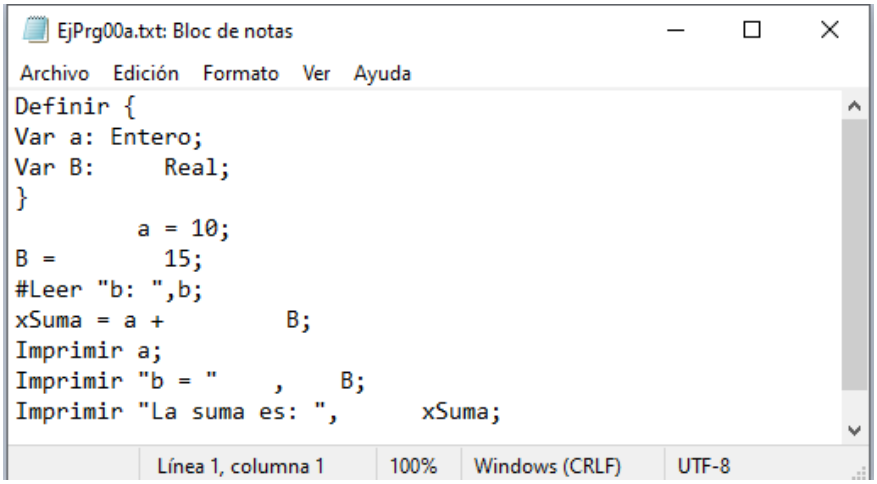
Cuando se trate de leer datos desde un archivo, se debe tomar en cuenta la ruta y el nombre del archivo con toda su extensión que siempre será ‘**txt**’.

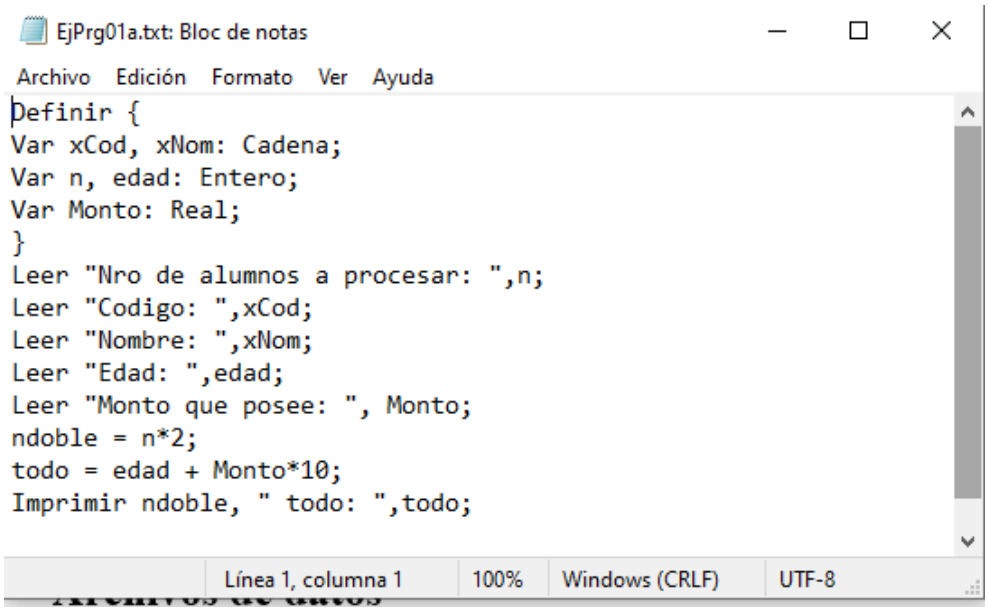
En mi caso, todos los archivos de programa fuente o de datos los guardo en la unidad de D y en la carpeta pyCcon, de manera que siempre he estado usando d:/pyCcon/nombreDelArchivo.txt

Presentamos aquí, imágenes de algunos archivos programa fuente:

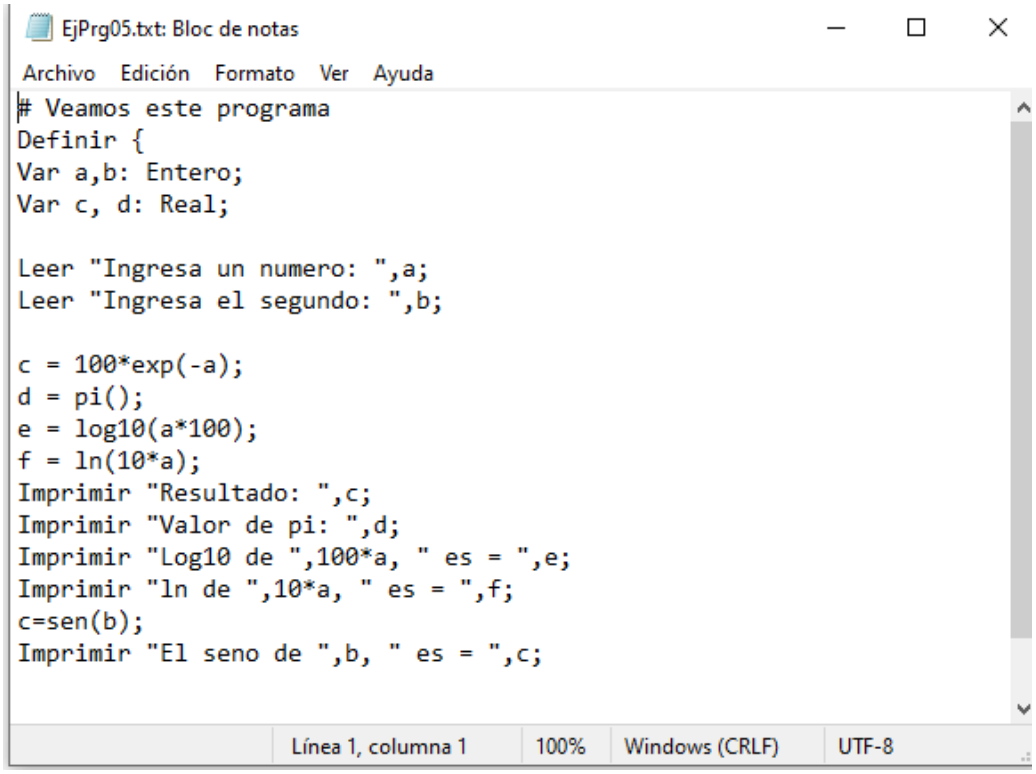



EjPrg01.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Definir {
Var X, Y: Vector;
}
a = 5;
b = -25;
xPago = 1250;
xTotalProducto = xPago + a*b
Línea 1, columna 1
100%
Windows (CRLF)
UTF-8

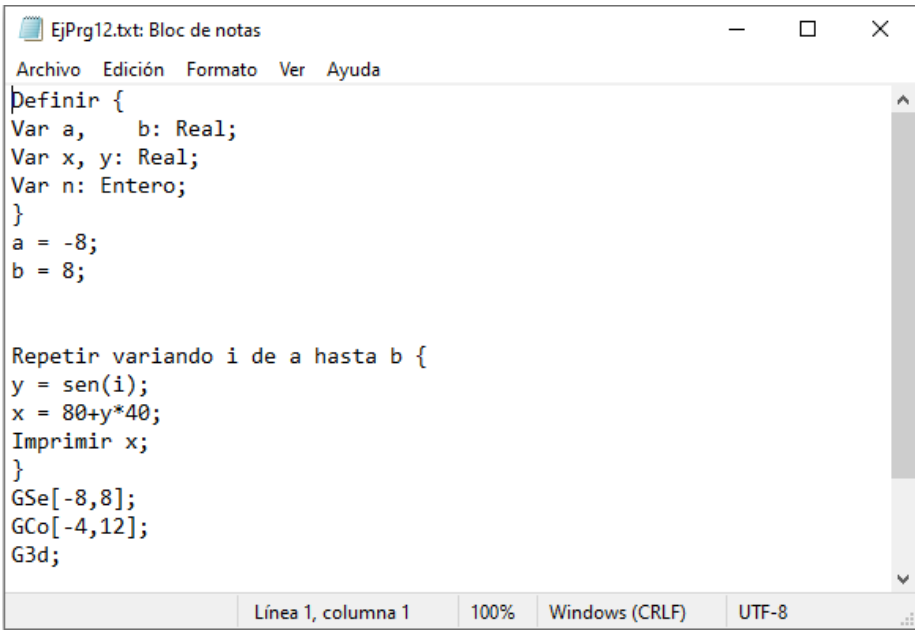

EjPrg00a.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Definir {
Var a: Entero;
Var B: Real;
}
a = 10;
B = 15;
#Leer "b: ",b;
xSuma = a + B;
Imprimir a;
Imprimir "b = " , B;
Imprimir "La suma es: ", xSuma;
Línea 1, columna 1
100%
Windows (CRLF)
UTF-8


EjPrg01a.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Definir {
Var xCod, xNom: Cadena;
Var n, edad: Entero;
Var Monto: Real;
}
Leer "Nro de alumnos a procesar: ",n;
Leer "Codigo: ",xCod;
Leer "Nombre: ",xNom;
Leer "Edad: ",edad;
Leer "Monto que posee: ", Monto;
ndoble = n*2;
todo = edad + Monto*10;
Imprimir ndoble, " todo: ",todo;
Línea 1, columna 1
100%
Windows (CRLF)
UTF-8

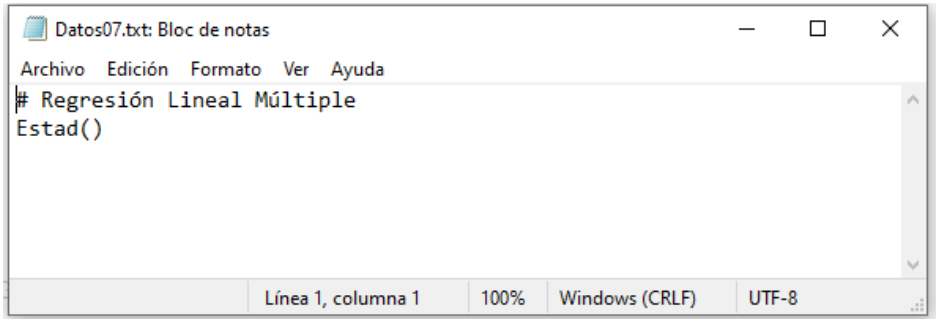
El siguiente es un archivo programa en el cual se están usando diversas funciones matemáticas explicadas anteriormente.



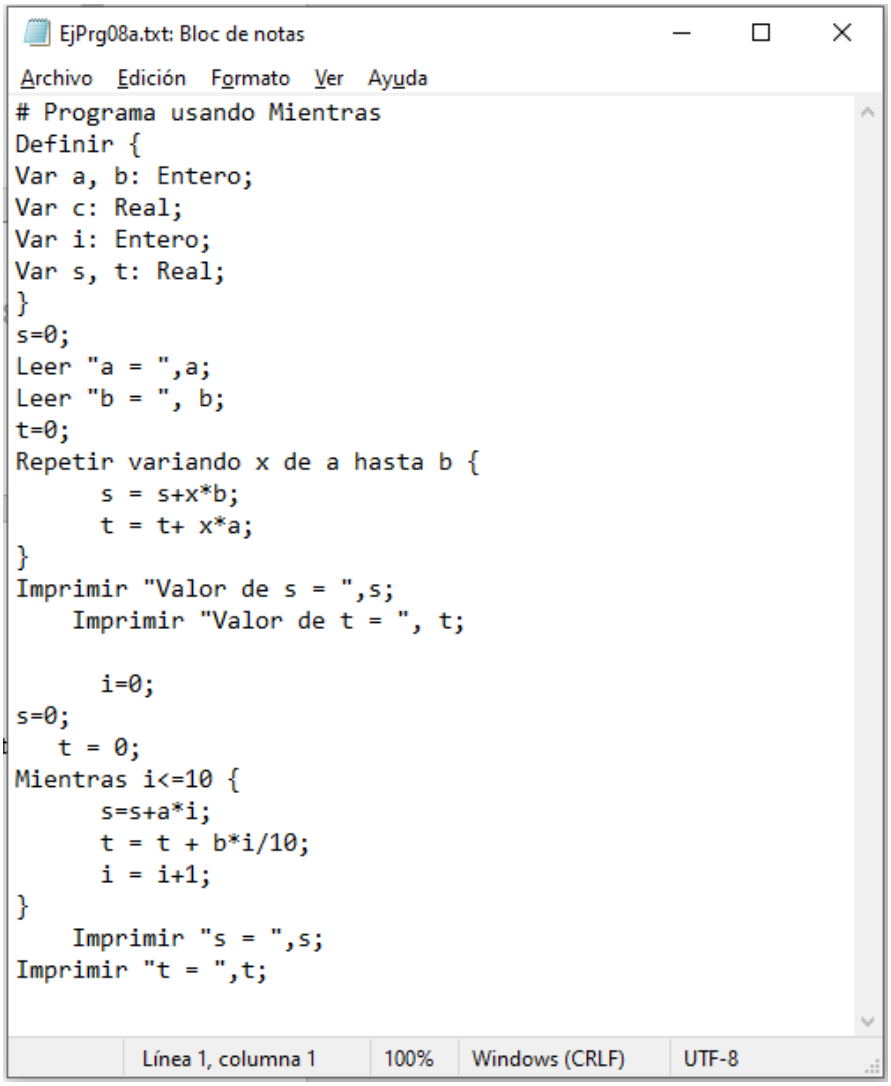
En el siguiente archivo tenemos un programa que realiza cálculos y traza gráficos de seno y coseno, así como construye un gráfico de superficie 3D.



Usaremos el siguiente archivo – programa para acceder a los diferentes ejemplos de regresión lineal mútlpile.



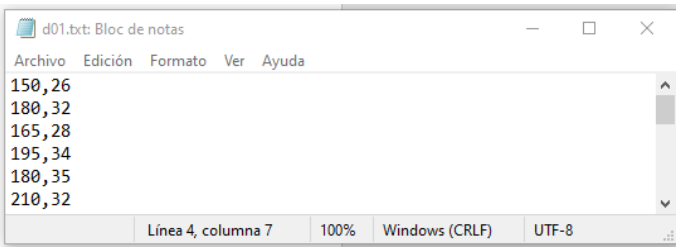
Finalmente presentamos un archivo en el cual se muestra diversas sentencias del lenguaje.

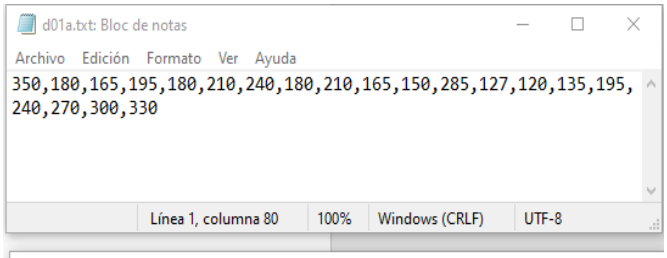


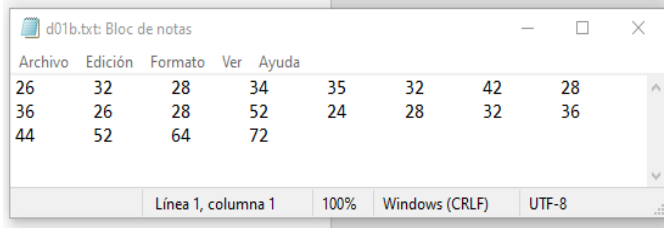
b. Archivos de datos

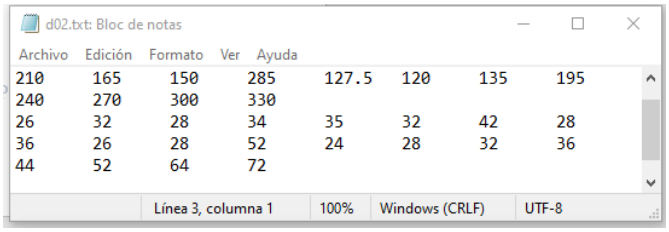
Así como en los archivos – programa hemos usado un editor de textos, de extensión txt, así también en la grabación de los archivos de datos usaremos ese tipo de archivo plano.

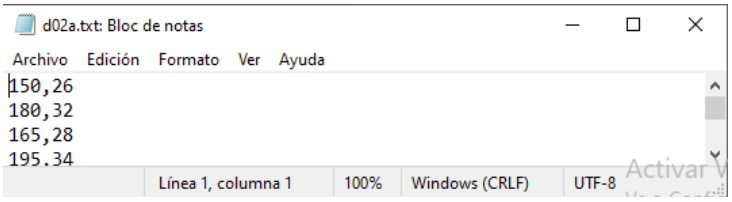
Mostramos algunos de los archivos usados:



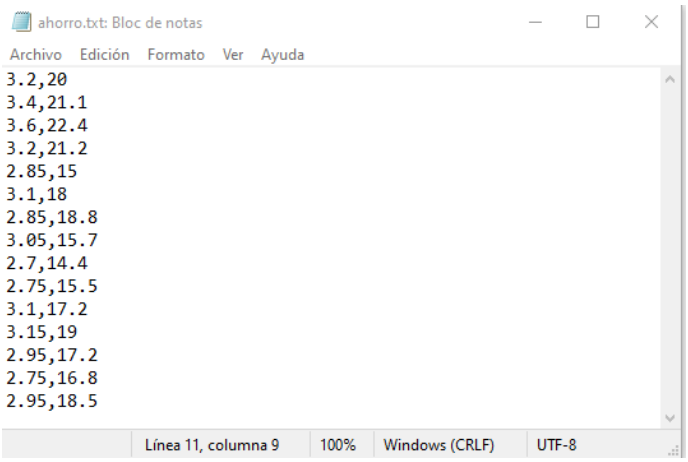


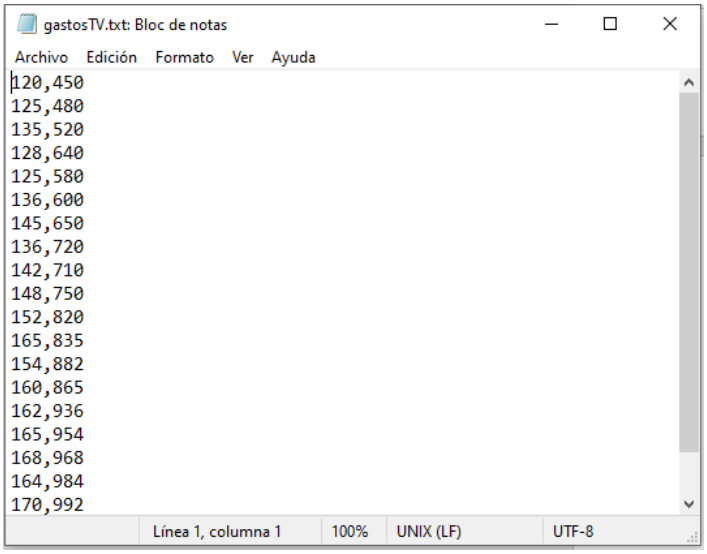


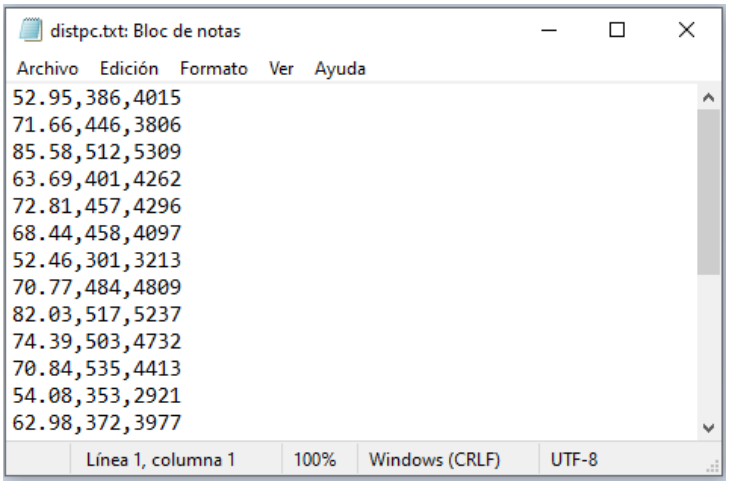




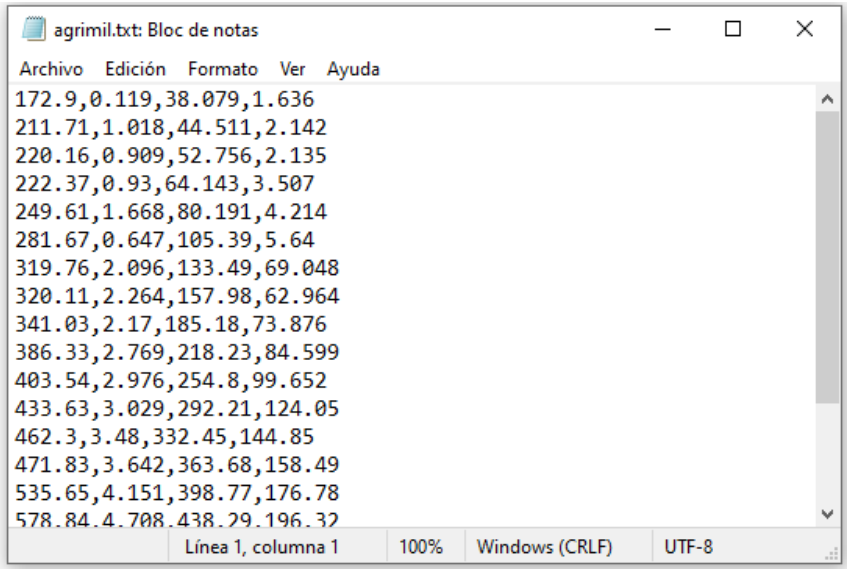
Para las aplicaciones de la Regresión Lineal Simple y Múltiple hemos usado los siguientes archivos de datos:







Y para una estimación de 4 variables:



c. Ejercicios

1. Escriba un programa llamado Ejer01.txt que imprima el mensaje: “Hola Mundo !!!”

2. Escriba un programa que permita leer desde el teclado tu nombre completo. El programa debe tener una primera línea de comentario: “Leer un dato” y la última Imprimir la variable leída.

3. Grabe un programa que permita leer dos números, que calcule en xSuma la suma de ellos y en xResta, su diferencia. Luego imprima los valores leídos y los calculados. Que sea el primer número como entero y al segundo como real.

4. Grabe un programa que lea un número real y luego obtenga su cuadrado, su raíz cuadrada, el cubo, la raíz cúbica, el logaritmo decimal y su logaritmo neperiano. Que los almacene en una cadena convertidos en caracteres.

   Que imprima todo.

5. Grabe un programa que permita leer un número entero n, que permita repetir n veces la suma del cuadrado, del cubo, de la raíz cúbica, del logaritmo decimal y del logaritmo neperiano, desde 1 hasta n. Luego imprima como en una tabla.

6. Escriba un programa que construya la gráfica de la función seno, del coseno y del seno y coseno en una misma gráfica, desde -8 hasta 8. Luego lea el valor de un ángulo e imprima una tabla desde -10 hasta 10 con incrementos de 2, del seno y coseno de ese valor.

7. Los ingresos por ventas de una tienda al menudeo durante los 12 meses del 2025 son las siguientes:

   45, 38, 54, 48, 66, 85, 120, 150, 132, 158, 180, 250

   Construya un gráfico de líneas y un gráfico de barras que permita visualizar las ventas mensuales.

8. Las utilidades reportadas por el Centro Comercial SoliMar durante los meses de Diciembre, enero, Febrero y Marzo, son las siguientes:

   3452.82, 6548.50, 12388.0, 5230.0

   Trace un gráfico de torta que permita visualizar las ventas durante el verano.

9. Dado el archivo ahorro.txt, donde se registra los ingresos y los ahorros de un trabajador, escriba un programa con el contenido mínimo de instrucciones para realizar un análisis de regresión lineal (son 3 líneas).

10.

ANEXOS

VII. ANEXO

1. ALGUNAS RESPUESTAS

b) Ejercicio 3:

```
Definir {
Var a: Entero;
Var b: Real;
}

Leer "Primer numero: ", a;
Leer "Primer numero: ", b;
xSuma = a+b;
xResta = a-b;
Imprimir "a = ", a, " b = ", b;
Imprimir "La suma: ",xSuma, "La resta: ",xResta;
```

c) Ejercicio 4

```
Definir {
Var x: Real;
}
cad = "";
Leer "Numero: ", x;
x2 = x*x;
x3 = pot(x,3);
x4 = raizc(x);
x5 = pot(x,(1/3));
x6 = log10(x);
x7 = ln(x);
cad = str(x)+' , '+str(x2)+' , '+str(x3)+' , '+str(x4)+' , ',str(x5)+' , '+str(x6)+' ,
'+str(x7);
Imprimir cad;
```

d) Ejercicio 5:

```
Definir {
# n será el número de repeticiones
Var n: Entero;
}

Leer "Numero de repeticiones: ", n;
sx2 = 0;
sx3 = 0;
srcub = 0;
slog = 0;
sln = 0;

Repetir variando i desde 5 hasta n {
sx2 = sx2 + i*i;
sx3 = sx3 + pot(i,3);
srcub = srcub + pot(i,(1/3));
slog = slog + log10(i);
sln = sln + ln(i);
cad = str(i)+' , '+str(sx2)+' , '+str(sx3)+' , '+str(srcub)+' , '+str(slog)+' ,
',str(sln);
Imprimir cad;
}
```

e) Ejercicio 6

```
Definir {
Var grados: Entero;
}
# Cálculos y gráficos sinusolidales
GSe[-8,8];
GCo[-8,8];
GSC[-8,8];
# Leer el ángulo
Leer "Ingrese el angulo: ",grados;
Repetir variando i de -10 hasta 10 incr 2 {
xs = sen(i);
xc = cos(i);
xcad = str(xs) + "    ,    " + str(xc);
Imprimir xcad;
}
```

f) Ddd
g) Fff

2. LISTA DE ARCHIVOS

El IceCompilador01.py, las librerías que usa, los archivos programa fuente escritos en LCON y los archivos de datos que hemos venido usando y otros, se encuentran almacenados en mi dominio

https://www.icepag.com

donde se encuentran alojados y desde el cual se pueden descargar.

Hay también otros programas escritos en Python y que pueden ser descargados y los que no los hemos usado pues el desarrollo del Python no ha sido nuestro objetivo sino en crear un nuevo lenguaje en español que al aprenderlo sirviera de intermediario para aprender de a pocos el Python.

Al descargar el compilador IceCompilador01.py debe colocarlo en una carpeta desde donde pueda cargarlo a memoria automáticamente. Recuerde que todo lo que hemos hecho es usando el compilador. Todos nuestros programas fuente en LCON, por pequeño que sea, se puede procesar ejecutando el compilador quien pide luego el programa fuente y después lo hará con los archivos de datos, si ese es el caso.

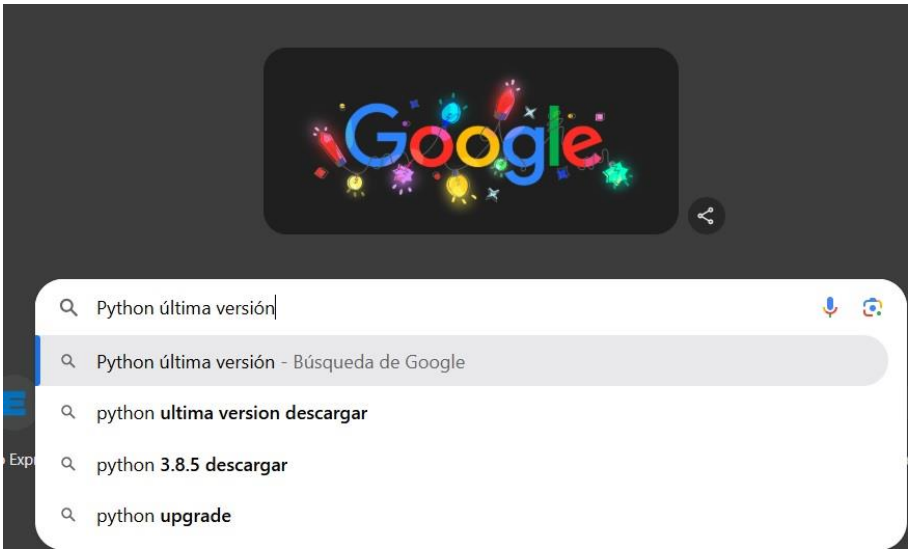
Las librerías que son partes del compilador se importan automáticamente. Las otras librerías del Python deben ser instaladas primero en el Shell del Windows para ser importadas por el compilador.

A continuación, tenemos el listado de todos estos archivos.

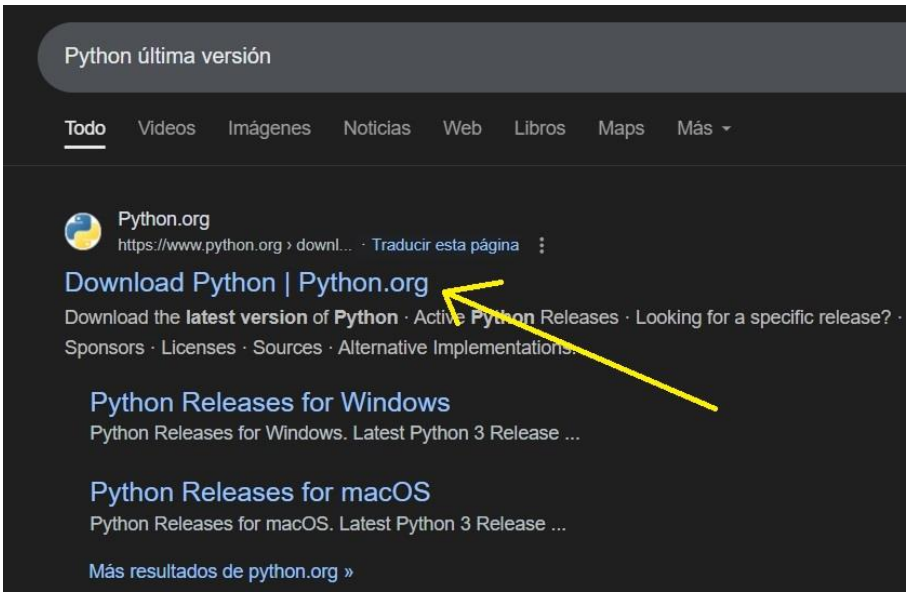
IceCompilador01.py
gr.Trig.py
gr.Funciones.py

3. INSTALACIÓN DE PYTHON

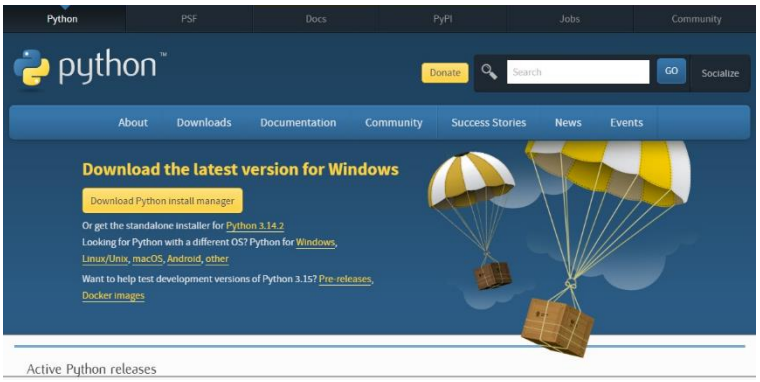
Paso 1: En la pantalla de Google digite:



Paso 2: Seleccionamos la dirección de Python.org

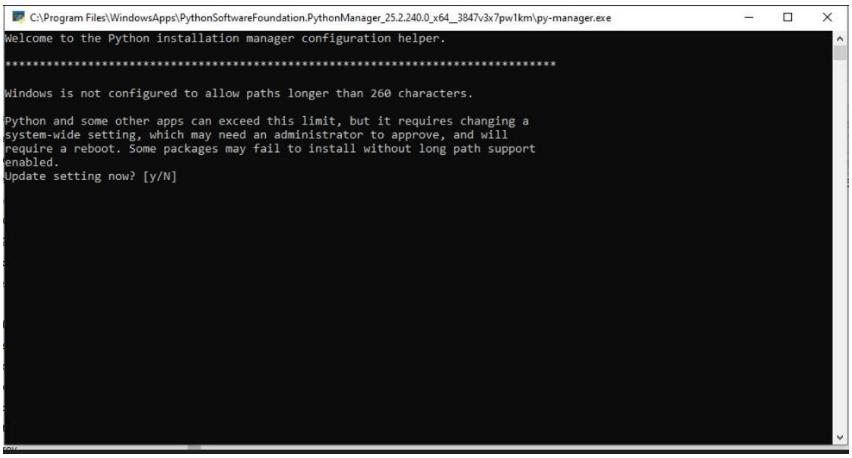


Paso 3: Seleccione descargar Python e instalar el administrador



Paso 4: Clic en Instalar Python

Paso 5: Cuando salga la ventana siguiente digite <y>

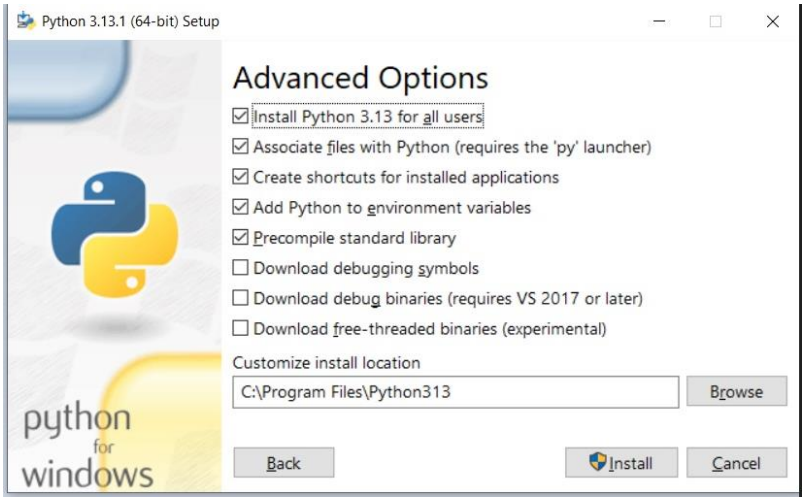


Paso 6:

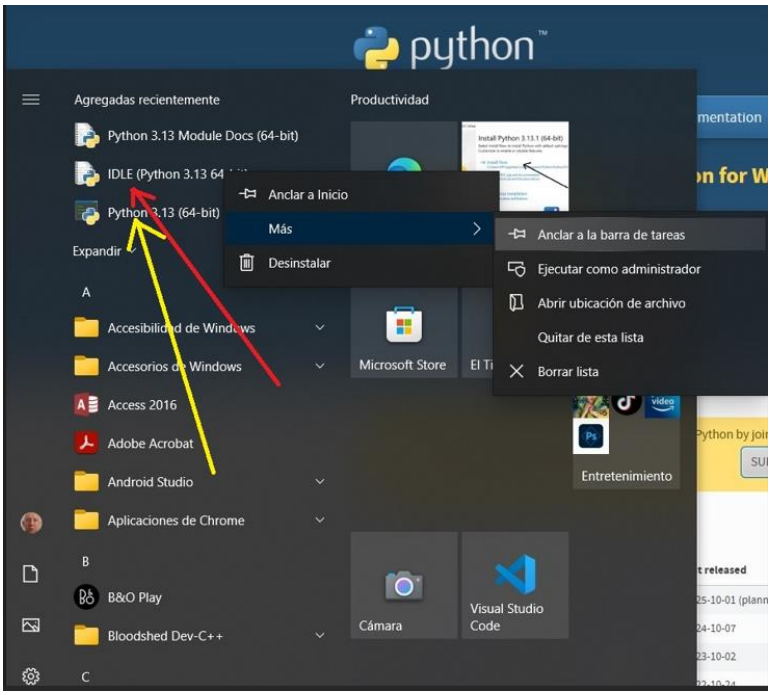
Continúe con la instalación paso a paso cuidando de activar las opciones de:

- <Install now>
- Instalar PIP
- Documentación

Paso 7: Tomar en cuenta opciones como ruta corta y variables de ambiente

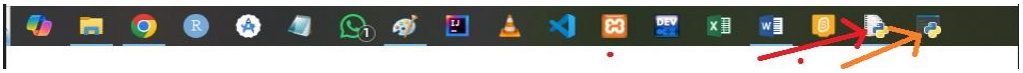


Paso 8: Luego de haber terminado la instalación activar la lista de las aplicaciones instaladas en la PC haciendo clic en el botón de inicio del Windows, donde se mostrará una entrada para Python que, al desplegarla se



observará opciones como se muestra en la imagen. Se debe seleccionar lo indicado por la flecha y colocarlo en la barra de tareas.

Paso 9: En la barra de tareas deben aparecer como se muestra aquí



La flecha roja nos muestra el ícono del IDLE del Python que, al hacer clic en él, se tendrá la ventana, el IDLE o el panel del Python que será el lugar donde se ingrese las órdenes y se reciba los resultados de todo lo que hagamos en Python. La otra flecha muestra el icono del Python que permite obtener una ventana en modo oscuro pero cuyo uso es muy restringido para ciertas operaciones.

4. INSTALACIÓN DE ALGUNAS LIBRERÍAS DEL PYTHON

Una vez instalado el Python, ya podemos usarla como una calculadora simple o compleja, el manejo de variables y de muchas operaciones estadísticas y matemáticas básicas. Al entorno del Python pertenecen ciertas funciones que conforman la librería que se instala con el Python como la librería random, la librería sys, math, etc.

Para disponer de ellas será suficiente importarlas usando el comando <Import> en el momento que se las necesite.

Para las siguientes operaciones que se requiera usar será necesario la instalación ciertas librerías que constituyen el gran aporte de los amantes del Python dentro de la filosofía del Sofware Libre.

Para disponer de ellas será necesario primero instalarlas para luego importarlas toda vez que se las necesite.

Para instalar estas librerías se debe usar el entorno del Shell del Windows para los cual, siga los siguientes pasos:

Paso 1: Use el botón derecho en el icono de <Inicio> del Windows
Paso 2: Haga clic en Windows Power Shell
Paso 3: En el prompt del Shell digite: **pip Install <nombre de la librería>**. Por ejemplo: **pip Install numpy**.

Si no se ejecuta o se obtiene error indicando que no está instalado el PIP, es más sencillo desinstalar el Python y volver a instalarlo nuevamente teniendo cuidado de activar la casilla donde se indica que se debe instalar el PIP.

Si se instala correctamente la librería entonces ya podemos cerrar la ventana del Shell y retornar al Python en donde se usará el comando <import>.

Por ejemplo, hemos instalado la librería del numpy. En el panel o IDLE ya podemos usar: **import numpy as np**.

Si hubiéramos instalado el matplotlib, en el IDLE o en algún módulo que se está creando, ya podemos digitar: **import matplotlib.pyplot as plt**.

5. BIBLIOGRAFÍA

De la mano con Python, Ilmer Cóndor E.
Estadística con Python, Ilmer Cóndor E.

No he necesitado ninguna otra información pues información y/o bibliografía del Lenguaje LCON, no existe, lo estoy creando.

Ilmer Cóndor E.