PRINCIPIOS BÁSICOS Y FUNDAMENTOS DE

PROGRAMACIÓN

Autores:

Maria Fernanda Medina Reyes
Maria Eugenia Rincón Socha

Universidad Tecnológica de Bolívar
CARTAGENA DE INDIAS

PRINCIPIOS BÁSICOS Y FUNDAMENTOS DE

PROGRAMACIÓN

Autores:

Maria Fernanda Medina Reyes
Maria Eugenia Rincón Socha

Universidad Tecnológica de Bolívar
CARTAGENA DE INDIAS

Medina Reyes, Maria Fernanda
Principios básicos y fundamentos de programación / Maria Fernanda Reyes Medina, Maria Eugenia Rincón Socha. - - Cartagena de Indias : Universidad Tecnológica de Bolívar, 2022.

ISBN: 978-628-7562-03-5 (digital)
65 páginas : Figuras y tablas

1.Python (lenguaje de programación de computadores) -- Manuales de laboratorio 2. Lenguajes de programación (computadores electrónicos) -- Manuales de laboratorio I. Medina Reyes, Maria Fernanda II. Rincón Socha, Maria Eugenia.

005.133
M491
CDD23

Primera edicción, julio 2022

Alberto Roa Varelo
Rector

Daniel Toro González
Vicerrector Académico

María del Rosario Gutiérrez de Piñeres
Vicerrectora Administrativa y Financiera

Javier Campillo Jimenez
Decano, Facultad Ingeniería

Jairo Useche Vivero
Director, de Investigación, innovación y emprendimiento

Edición
Editorial UTB

ISBN: 978-628-7562-03-5 (digital)

Campus Casa Lemaitre: Calle del Bouquet
Cra 21 No 25-92 PBX (5) 6606041 -42- 43 Fax: (5) 6604317
Campus Tecnológico:

Parque Industrial y Tecnológico Carlos Vélez Pombo
PBX (5) 6535331 Fax: (5) 6619240
Cartagena - Colombia

www.utb.edu.co

Contenido

Introducción

La programación consiste en resolver problemas creando soluciones paso a paso escritas en un lenguaje específico. El lenguaje de programación Python se utiliza por su facilidad de uso debido a sus altas capacidades, siendo una buena herramienta para introducir conceptos y técnicas básicas (Liang, 2013). Además, Python proporciona bibliotecas para realizar aplicaciones avanzadas que facilitan el manejo de problemas de matemáticas, ciencias sociales, finanzas, administración de empresas, juegos y multimedia, etc.

Esta guía de ejercicios usa el lenguaje de programación Python para practicar los conceptos fundamentales de programación, en el cual se resuelven problemas básicos de ingeniería siguiendo las instrucciones dadas de acuerdo con los ejercicios propuestos. Las herramientas de trabajo en las cuales se pueden desarrollar los ejercicios son de libre uso facilitando así el acceso al entorno de trabajo para la realización de cada práctica.

Los temas de cada práctica han sido seleccionados para complementar la parte teórica de los fundamentos de programación abordando los temas de estructura secuencial, condicional, repetitiva y fundamentos de estructura de datos, quedando la temática de la presente guía así organizada:

- Estructura Secuencial – E – P – S
- Estructura Condicional if – elif – else
- Estructura Repetitiva while – for
- Estructura de datos – Fundamentos

Para finalizar, cada ejercicio tendrá datos de prueba donde se evalúa la funcionalidad del ejercicio realizado. Los datos de prueba son útiles para verificar el resultado esperado de la solución propuesta, de tal forma que permita fortalecer así las habilidades de programación a través de diferentes situaciones y garantice la funcionalidad correcta del programa.

Objetivos y metodología de los laboratorios

Objetivo general del laboratorio

Realizar prácticas de fundamentos de programación, mediante el uso de herramientas de acceso libre y a través de un lenguaje de programación Python, con el fin de afianzar los conocimientos teóricos básicos de programación.

Objetivos específicos del laboratorio

- Satisfacer las inquietudes que tenga una persona que está dando sus primeros pasos en programación respecto la aplicación de los principales temas de fundamentos de programación, tales como: estructura secuencial, condicional, repetitiva y fundamentos de estructuras de datos.
- Motivar la experiencia en aspectos como: manejo del laboratorio virtual de programación, diseño, creación y ejecución de un programa con las diferentes pruebas que se presentan.
- Fomentar el interés general por las habilidades de la lógica computacional y la programación en Python.

Metodología recomendada

Metodológicamente se plantean prácticas y ejercicios de autoevaluación, los cuales están agrupados en los siguientes temas: Estructura Secuencial E – P – S, Estructura Condicional if – elif – else, Estructura Repetitiva while – for, y Estructura de datos – Fundamentos.

Cada ejercicio está integrado (estructurado) por los siguientes elementos: título, objetivo de la actividad, competencias para desarrollar o fortalecer, enunciado, penalizaciones, descripción de los datos de entrada, descripción de los datos de salida y datos de prueba. A continuación se describen estos apartados:

Título: Corresponde al nombre de la práctica, este título está relacionado con el tema a desarrollar.

Objetivo de la actividad: Determina los logros que se deberán demostrar al finalizar cada laboratorio.

Competencias para desarrollar o fortalecer: Son las competencias y habilidades que se esperan cuando se finalice el laboratorio de programación.

Enunciado: Expresa la situación de manera concreta con todas las especificaciones del caso y describe exactamente cuál es el objetivo que se quiere lograr. El interesado deberá dedicar suficiente tiempo para la comprensión del enunciado y estructurar posibles soluciones,

teniendo en cuenta los requerimientos funcionales y el cómo hacer para obtener la salida a partir de los datos de entrada. La siguiente grafica orienta en la planeación de su solución. En esta etapa se formalizan las etapas de solución del problema (entrada, proceso y salida E – P – S) y se organizan con el criterio Top - Down (de arriba hacia abajo).

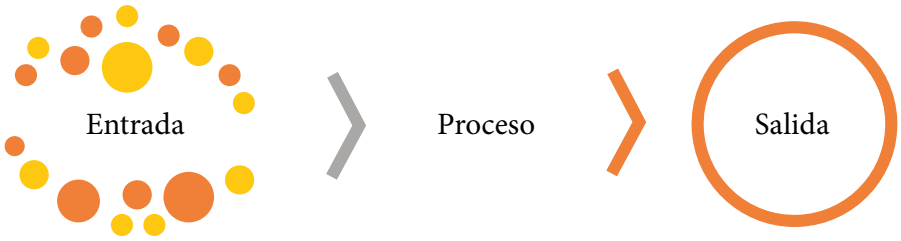


Figura 1. Elementos de un programa: Entrada - Proceso - Salida

Descripción de los datos de entrada: Son los datos que se ingresan al programa, los cuales por lo general, son ingresados mediante teclado. La denominación en programación para un dato de entrada es *input*.

Descripción de los datos de salida: Se determinan las salidas, escribiendo de manera explícita cada una de ellas, mediante el lenguaje de programación. La denominación en programación para un dato de salida es *print*.

Datos de prueba: Permiten verificar la eficacia del algoritmo y dependiendo de los resultados que se vayan obteniendo durante la práctica debe decidir qué acciones tomar, en caso de que haya errores.

Conclusiones: Facilita al interesado reflexionar sobre el aprendizaje adquirido en la práctica o actividad realizada.

1. Entornos de programación de acceso libre

Existen muchas herramientas de acceso libre donde se puede verificar que un programa está funcionando y normalmente la forma de acceder es mediante un breve registro. A continuación se describen algunas herramientas online donde se puede crear y guardar cada una de las prácticas presentadas en este documento.

1.1. replit

Es un entorno de desarrollo integrado para trabajar desde el navegador cuyo enlace de acceso es https://replit.com/. Es una herramienta de fácil uso, donde inicialmente es necesario seleccionar el lenguaje de programación, en este caso es Python. Como observamos en la figura 2, en el campo de **Title** se recomienda escribir el nombre del proyecto lo cual nos permitirá identificar posteriormente los trabajos por el nombre. Seguidamente se presiona el botón **Create Repl** que se muestra resaltado en color azul.

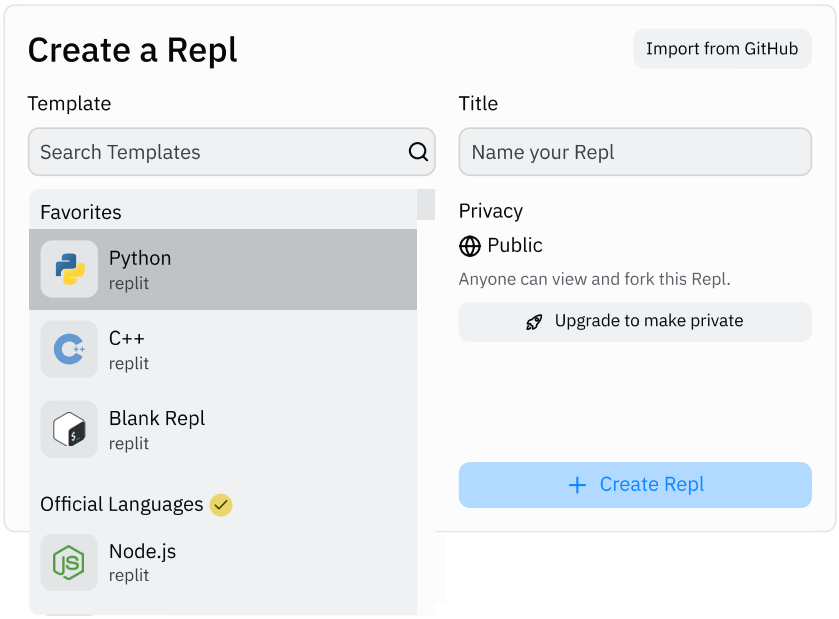


Figura 2. Creación de un programa en replit

En la siguiente captura de pantalla se puede observar un ejemplo de un programa realizado en replit. Para ejecutarlo se hace clic en el botón verde que dice **Run** y en el recuadro en negro donde dice **Console** se mostrará el resultado de esta ejecución, en este caso se presentaría la palabra *Hola Mundo*.

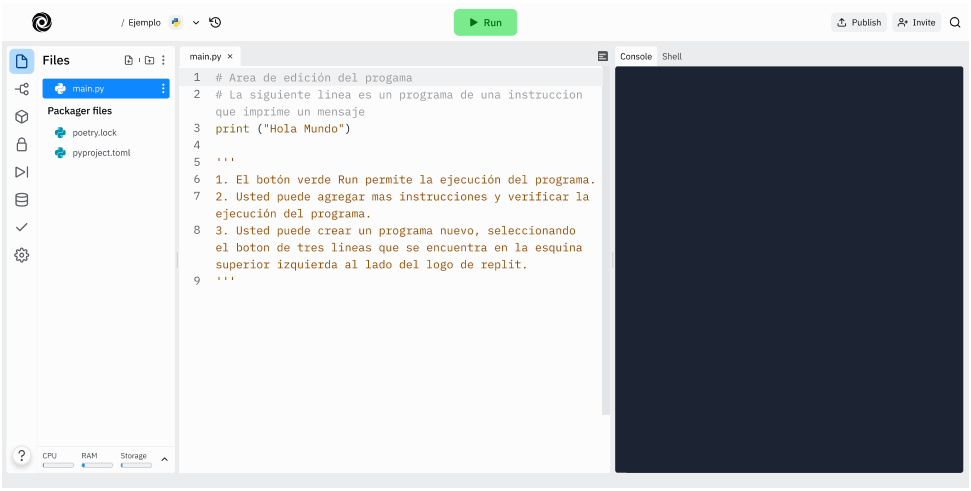


Figura 3. Ejemplo de un programa en replit

1.2. Google Colab

Es una herramienta de trabajo de Google que permite trabajar colaborativamente cuyo enlace de acceso es https://colab.research.google.com/. Se recomienda seleccionar un **Nuevo Notebook**[1] para la creación de un proyecto como se muestra en la siguiente figura:

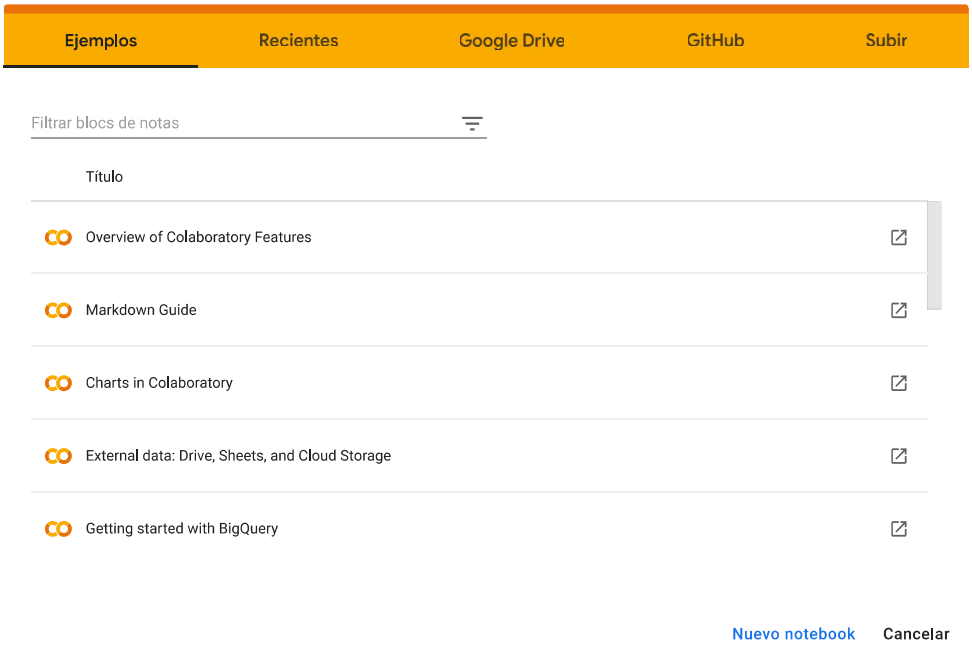


Figura 4. Creación de un Notebook en Google Colab

[1] Notebook es un cuaderno de trabajo de programación en el cual se escribe código en celdas que se pueden ejecutar.

Para cambiar el nombre del proyecto se hace clic en **cambiar nombre del cuaderno** que se encuentra en la parte superior izquierda y se escribe un nuevo nombre, en este caso se ha puesto el nombre de “ejemplo.ipynb”. Como se observa en la siguiente figura se puede añadir una celda con **Código** o se puede insertar una celda con **Texto** y para ejecutar las instrucciones se hace clic en el botón ejecutar que se encuentra en el lado izquierdo de cada celda. El botón ejecutar tiene forma de círculo negro con un triángulo inscrito en él.

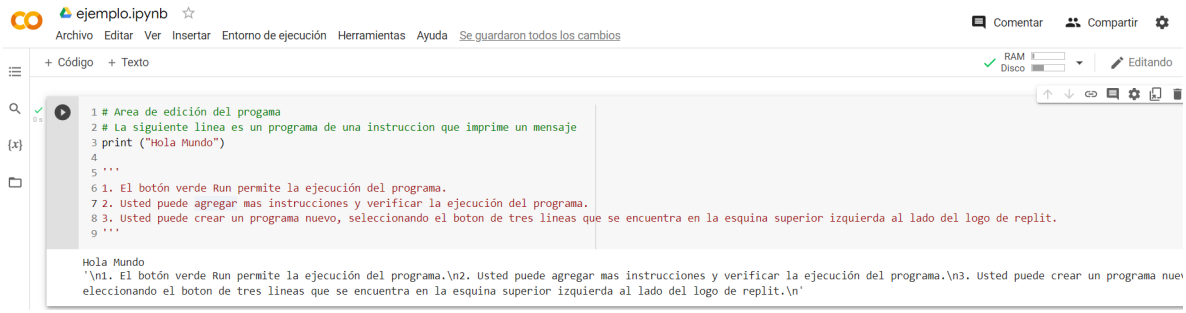


Figura 5. Ejemplo de un Notebook en Google Colab

2. Estructura Secuencial – E – P – S

2.1. Laboratorio raíz cuadrada y potencia de un número

Objetivos de la actividad

- Resolver ejercicios secuenciales identificando Entrada – Proceso – Salida.
- Comprender y aplicar la estructura básica de un programa en Python.
- Aplicar las diferentes alternativas para realizar el proceso de un algoritmo secuencial, utilizando variables, expresiones aritméticas y operaciones de asignación.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Identificar la estructura de un algoritmo y el rol que juega cada uno de los elementos que lo componen.
- Ser propositivo en la búsqueda de alternativas para la solución de problemas.

Enunciado

Desarrolle un algoritmo, que dado un número ingresado por teclado retorne su raíz cuadrada, y el cuadrado del número. Los datos ingresados por teclado deben ser de tipo real (**float**). Usted debe usar los diferentes métodos para el cálculo de la raíz cuadrada y para la potencia de un número. Utilice la función **round** para redondear a dos cifras decimales los resultados.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 80% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Entrada	Tipo de dato	Descripción
numero	Numérico (real)	Número para evaluar que se ingresa por teclado.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
raiz	Numérico (real)	Raíz cuadrada del número ingresado. Redondeado a dos cifras decimales.
potencia	Numérico (real)	Potencia del número ingresado. Redondeado a dos cifras decimales.

Datos de prueba (Los datos de entrada y salida son en el orden que se muestra)

Entrada	Salida
numero	**print ( )**
5	"La raiz de **5.0** es **2.24**" "La potencia de **5.0** es **25.0**"
29	"La raiz de **29.0** es **5.39**" "La potencia de **29.0** es **841.0**"

Se recomienda que la función que usted cree tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def potenciaRaiz():

    # Desarrolle aquí su algoritmo

    pass

# Llamado de la función
potenciaRaiz()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

2.2. Laboratorio suma de dos números

Objetivos de la actividad

- Resolver ejercicios secuenciales identificando Entrada – Proceso – Salida.
- Comprender y aplicar la estructura básica de un programa en Python.
- Aplicar las diferentes alternativas para realizar el proceso de un algoritmo secuencial, utilizando variables, expresiones aritméticas y operaciones de asignación.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Identificar la estructura de un algoritmo y el rol que juegan cada uno de los elementos que lo componen.
- Ser propositivo en la búsqueda de alternativas para la solución de problemas.

Enunciado

Desarrolle un algoritmo, que dado dos números (**float**) ingresados por teclado imprima su suma. El resultado debe estar redondeado a dos cifras decimales.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 80% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Entrada	Tipo de dato	Descripción
numero1	Numérico (real)	Primer número ingresado por teclado.
numero2	Numérico (real)	Segundo número ingresado por teclado.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
suma	Numérico (real)	Resultado de la suma redondeado a dos cifras decimales.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

numero1	numero2	Salida – print ( )
2.0	5.0	"la suma de **2.0** + **5.0** es **7.0**"
-5.0	-8.0	"la suma de **-5.0** + **-8.0** es **-13.0**"

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación

'''
Escriba aquí las conclusiones
'''

def sumaDeDosNumeros():

    # Desarrolle aquí su algoritmo
    pass

# Llamado de la función
sumaDeDosNumeros()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

2.3. Laboratorio área de un polígono

Objetivos de la actividad

- Convertir una expresión algebraica a expresión algorítmica, haciendo uso de la jerarquía de las operaciones.
- Utilizar la librería matemática como apoyo para los cálculos de las funciones trigonométricas.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser propositivo en la búsqueda de alternativas para la solución de problemas.
- Ser organizado en la manipulación de las variables y los diferentes tipos de datos.

Enunciado

Escriba un algoritmo que permita calcular el área de un polígono determinado, el usuario solo ingresa el número de lados y la longitud de un polígono regular de tipo de dato real (**float**) y como resultado se muestra el área con 4 cifras decimales.

La fórmula para calcular el área de un polígono regular es:

$$Area = \frac{n \times s^2}{4 \times \tan\left(\frac{\pi}{n}\right)}$$

Donde,

s, es la longitud de un lado y **n**, es el número de lados.

El usuario solo ingresa el número de lados y su longitud para calcular y mostrar el área como salida.

Nota: Recuerde usar la librería matemática (**math**) para importar la función trigonométrica (**tan**) y obtener el valor de π (**pi**).

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 80% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Entrada	Tipo de dato	Descripción
s	Numérico (real)	Longitud de un lado.
n	Numérico (real)	Número de lados del polígono.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
area	Numérico (real)	área calculada redondeado a 4 cifras decimales.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Entrada		Salida
n	**s**	**print ( )**
5	6.5	"El área calculada es **72.6902**"
6	5.5	"El área calculada es **78.5918**"

Se recomienda que la función creada por tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def calculoAreaPoligono():

    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
calculoAreaPoligono ()
```

Conclusiones

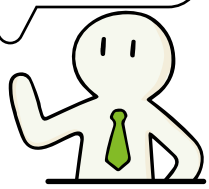
Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

2.4. Ejercicios de autoevaluación de Estructura Secuencial

Ejercicio 1. Introducción a algoritmos

Algoritmo: Es un conjunto de instrucciones bien definidas y ordenadas lógicamente para llegar a la solución de un problema. Ejemplo: Algoritmo para prender un computador (Joyanes, 2008).

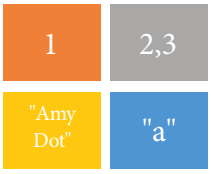
Palabras claves:
- Algoritmo
- Partes de un programa



Ordena los pasos para prender un computador:

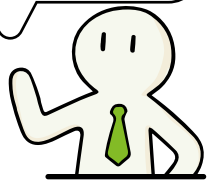
	Desorden	Orden
1	Encender pantalla	
2	Encender estabilizador	
3	Conectar la pantalla	
4	Conectar la torre	
5	Listo	
6	Conectar la pantalla	

Ejercicio 2. Tipos de datos



Un tipo de dato: Es un atributo de una parte de los datos que indica al ordenador (y/o al programador) algo sobre la clase de datos sobre los que se va a procesar. En Python encontramos int, float, str (string) y bool (Liang, 2013).

Palabras claves:
- Tipos de dato



Selecciona el tipo de dato correspondiente a cada valor dado:

1) 431 es un tipo de dato:

a) int b) float c) str d) bool

2) "Esteban" es un tipo de dato:

a) int b) float c) str d) bool

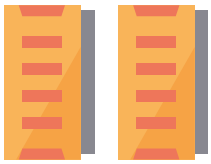
3) 'a' es un tipo de dato:

a) int b) float c) str d) bool

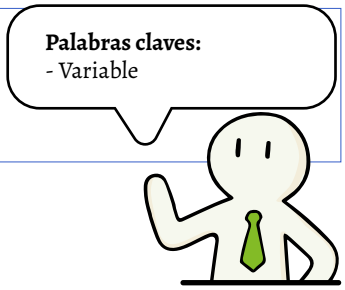
4) 15.1 es un tipo de dato:

a) int b) float c) str d) bool

Ejercicio 3. Variables y tipos de datos



Una variable representa a un espacio de memoria en el cual se puede almacenar un dato (Joyanes, 2008)



Selecciona la respuesta correcta a las siguientes preguntas:

1) ¿Cómo declaro una variable llamada **edad** que contiene la edad de una persona?

a) edad = input ("Edad: ") b) edad = int (input ("Edad: "))

c) edad = float (input ("Edad: ")) d) input ("Edad: ")) = edad

2) ¿Cómo declaro una variable llamada **nombre** que contiene el nombre de una persona?:

a) nombre = input ("Nombre: ") b) nombre = float (input ("Nombre: "))

c) nombre = Nombre d) input ("Nombre: ")

3) Ordena por orden de prioridad los siguientes operadores:

Prioridad	Operadores
	potencia (**),*, /, %
	(+ positivo), (- negativo)
	( )
	+, -

4) Calcula el resultado de las siguientes expresiones:

a. 8 + 7 * 3 + 4 * 6 b. 3 + 2 * (18 – 4 ** 2) c. 16 * 6 – 3 * 2

5) Selecciona el valor correcto de la variable **y** después de evaluar (ejecutar) las siguientes expresiones:

```
a = 4 * 8
b = a ** 2
y = ( a + b ) / 2
```

a) 32 b) 64 c) 528 d) 321

Ejercicio 4 . Solución de problemas

Analizamos un problema teniendo en cuenta lo siguiente: entender el problema, conocer el resultado, identificar datos e información, definir las operaciones.

Selecciona la respuesta correcta:

```
a = 5
b = 3
suma = a + b
print ('La suma es: ', suma)
```

1) ¿Qué realiza el anterior algoritmo?

a. Suma de dos números Reales
b. Suma de tres números Enteros
c. Suma de dos números Enteros
d. Resta de dos números Enteros

2) ¿ Cuantas variables tenemos?

a. Tres variables de tipo Entero
b. Dos variables Entero y una Real
c. Una variable Real llamada suma
d. Ninguna de las anteriores

3) ¿Qué valor guardará la variable suma?

a. 8 b. 5 c. 6 d. 9

4) ¿Cuál es la salida del algoritmo?

a. Suma de a y b
b. Declarar a y b como Entero
c. Declarar suma como Entero
d. Valores de 5 y 3

3. Estructura Condicional if – elif – else

3.1. Laboratorio comisión de un vendedor

Objetivos de la actividad

- Diseñar correctamente una estructura condicional.
- Identificar la sentencia condicional a usar y distinguir la estructura simple, doble, múltiple o anidada.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Usar alternativas para la solución de problemas.
- Ser creativo al seleccionar las instrucciones de selección.

Enunciado

Un vendedor de accesorios y perfumes recibe un sueldo base, adicional a este sueldo recibe un porcentaje (10%) extra por comisiones de sus ventas. El vendedor desea saber cuánto dinero obtendrá si realiza tres (3) ventas en el mes, y el total que recibirá en el mes, tomando en cuenta su salario más las comisiones. Si el total de las comisiones sobrepasan un monto de $200.000 pesos, se debe imprimir un mensaje donde se indique si el vendedor se hace acreedor a un obsequio.

Diseñe un algoritmo que calcule y muestre el total a pagar de comisiones para el vendedor, calcule y muestre el total a pagar en el mes al vendedor, y para finalizar, muestre un mensaje si el vendedor se hizo o no acreedor a un obsequio, teniendo en cuenta que la condición es que el monto sea superior a $200.000 en comisiones.

Se debe pedir como dato de entrada el sueldo base (**float**), venta1 (**float**), venta2 (**float**) y venta3 (**float**), y a partir de estos datos realizar los cálculos necesarios para mostrar en la salida. La respuesta debe ser expresada con dos cifras decimales en todos los casos.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior

a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
sueldoBase	Numérico real (float)	Salario base que recibe el trabajador sin comisiones.
venta1	Numérico real (float)	Monto de la venta número 1.
venta2	Numérico real (float)	Monto de la venta número 2.
venta3	Numérico real (float)	Monto de la venta número 3.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
comisiones	Numérico real (float)	Total a pagar de comisiones para el vendedor. Redondeado a dos cifras decimales.
totalMes	Numérico real (float)	Total a pagar en el mes al vendedor. Redondeado a dos cifras decimales.
obsequio	Cadena de texto (str)	Mensaje que se muestra si el vendedor alcanza o no alcanza un monto superior a $200.000 en las comisiones. El mensaje debe decir **"No recibe obsequio"** o **"Si recibe obsequio"**.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Recuerde que al ingresar los datos de entrada, no se deben ingresar los separadores de mil.

Datos de entrada				Datos de salida		
sueldoBase	**venta1**	**venta2**	**venta3**	**comisiones**	**totalMes**	**obsequio**
$ 1.850.000	$ 300.000	$ 450.000	$ 250.000	"Total comisión: 100000.0"	"Total mes: 1950000.0"	"No recibe obsequio"
$ 1.750.000	$ 500.000	$ 500.000	$ 145.000	"Total comisión: 114500.0"	"Total mes: 1864500.0"	"No recibe obsequio"
$ 1.850.000	$ 519.000	$ 680.000	$ 801.001	"Total comisión: 200000.1"	"Total mes: 2050000.1"	"Si recibe obsequio"

Se recomienda que la función que usted cree tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''


def calculoComisiones():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función

calculoComisiones()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

3.2. Laboratorio evaluación de un ángulo positivo en cuadrantes del plano cartesiano

Objetivos de la actividad

- Diseñar correctamente una estructura condicional.
- Identificar la sentencia condicional a usar y distinga las estructuras simple, doble, múltiple o anidada.

Competencias para desarrollar o fortalecer

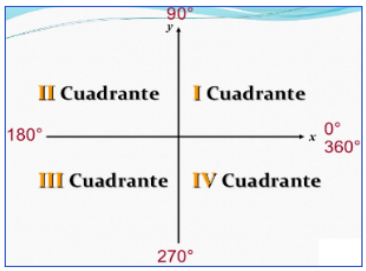
Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser proactivo al seleccionar expresiones relacionales y lógicas.
- Ser creativo al seleccionar las instrucciones de selección.

Enunciado

El cuadrante en el que reside una línea trazada desde el origen es determinado por el ángulo que forma la línea con el eje x positivo como sigue:

Ángulo desde el eje x positivo	Cuadrante
Entre 0 y 90 grados	I
Entre 90 y 180 grados	II
Entre 180 y 270 grados	III
Entre 270 y 360 grados	IV



Usando esta información, escriba un algoritmo que acepte el ángulo en grados como dato de entrada y determine el cuadrante apropiado de los datos introducidos. Si el ángulo tiene exactamente 0, 180 o 360 grados, la línea se encuentra en el eje x, y si el ángulo tiene exactamente 90 o 270 grados la línea se encuentra en el eje y.

Nota: Los ángulos a evaluar en el algoritmo son positivos, en caso contrario se debe mostrar el mensaje "**Dato erróneo**".

Se debe pedir como dato de entrada el ángulo (**float**), a partir de este, realizar los cálculos necesarios para mostrar la información solicitada del ángulo.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
angulo	Numérico real (float)	Ángulo para evaluar.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
mensaje	Cadena de texto (str)	Cuadrante apropiado a los datos introducidos: "**Cuadrante I**", "**Cuadrante II**", "**Cuadrante III**" o "**Cuadrante IV**". Si el ángulo tiene exactamente 0, 180 o 360 grados, la línea se encuentra en el "**Eje x**", y si el ángulo tiene exactamente 90 o 270 grados la línea se encuentra en el "**Eje y**". Nota: Los ángulos a evaluar en el algoritmo son positivos y hasta 360 grados, en caso contrario se debe mostrar el mensaje "**Dato erróneo**".

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra):

Datos de entrada	Datos de salida
angulo	**print ( )**
45	"Cuadrante I"
0	"Eje x"
-360	"Dato erróneo"

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def cuadrante():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
cuadrante()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

3.3. Laboratorio monto de comisiones

Objetivos de la actividad

- Diseñar correctamente una estructura condicional.
- Identificar la sentencia condicional a usar y distinga las estructuras simple, doble, múltiple o anidada.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser proactivo al seleccionar expresiones relacionales y lógicas.
- Ser creativo al seleccionar las instrucciones de selección.

Enunciado

En una tienda de ropa se desea calcular el monto de las comisiones que recibirá un vendedor, cuyos datos obtenidos son: su nombre (**str**), total de unidades que vendió (**int**) y el precio del artículo vendido (**float**).

A partir de esta información se debe calcular el monto de dicha comisión de la siguiente forma: Si el precio del artículo es de **$20.000** o menos, la comisión es del **3%**, si el precio del artículo es mayor que **$20.000** pero menor que **$50.000** la comisión será del **5%**, si el precio del artículo es mayor o igual que **$50.000** la comisión será del **10%**.

Diseñe un algoritmo que muestre el título del programa "**MONTO DE COMISIONES**" (**str**), calcule y muestre también el nombre del vendedor (**str**) junto con el total a pagar de comisiones (**float**).

Se debe pedir como dato de entrada el nombre del vendedor (**str**), la cantidad de unidades vendidas (**int**), y el precio del artículo (**float**). Y, a partir de estos datos, realizar los cálculos necesarios para mostrar el monto de comisiones que recibirá un vendedor.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	**Tipo de dato**	**Descripción**
titulo	Cadena de texto (str)	Este dato solo es un mensaje (**print**) que indica el nombre del programa. El texto debe ser igual a "**MONTO DE COMISIONES**". Este mensaje debe ir antes de todo el programa.
nombre	Cadena de texto (str)	Nombre del vendedor.
unidades	Numérico entero (int)	Cantidad de unidades vendidas.
precio	Numérico real (float)	Precio del artículo.

Descripción de los datos de salida

Salida	**Tipo de dato**	**Descripción**
nombre	Cadena de texto (str)	Nombre del vendedor.
comisiones	Numérico real (float)	Total a pagar de comisiones.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Datos de entrada				Datos de salida
titulo – print ( )	nombre	unidades	precio	print ( )
"MONTO DE COMISIONES"	"Pedro"	2	$5.000	"**Pedro** obtiene una comisión de **300.0**"
"MONTO DE COMISIONES"	"Carlos"	7	$20.000	"**Carlos** obtiene una comisión de **4200.0**"
"MONTO DE COMISIONES"	"Ernesto"	4	$25.000	"**Ernesto** obtiene una comisión de **5000.0**"

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def montoComisiones():
    # Desarrolle aquí su algoritmo

    pass
# Llamado de la función
montoComisiones()
```

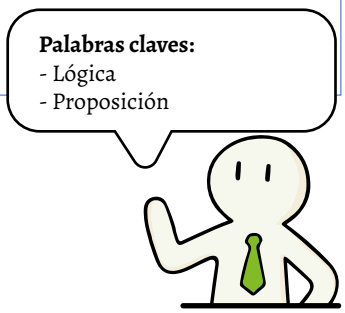
Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

3.4. Ejercicios de autoevaluación de Estructura Condicional

Ejercicio 1. Lógica proposicional

Las proposiciones son afirmaciones verdaderas o falsas, pero no ambas. Ejemplo:
"Cinco más siete son doce" es Verdadero.
"Cinco más siete son trece" es Falso.



Responda Verdadero o Falso

1) La suma de 3 más 1 es 4

a) Verdadero b) Falso

2) El año tiene 266 días

a) Verdadero b) Falso

3) La capital de Colombia es Cartagena

a) Verdadero b) Falso

4) El triángulo tiene tres lados

a) Verdadero b) Falso

5) La recta es una curva

a) Verdadero b) Falso

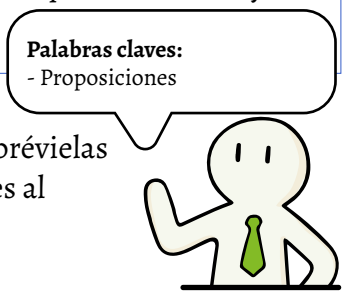
6) 2 es un número impar

a) Verdadero b) Falso

Ejercicio 2. Interpretación del lenguaje simbólico a lenguaje natural

Los conectivos Lógicos permiten construir proposiciones compuestas. En Python encontramos los operadores lógicos and, or y not. En el siguiente ejemplo se tienen dos proposiciones P y Q: P: Llueve y Q: Hace Sol

En el cálculo proposicional se escribe: P ∧ Q, esto equivale a Llueve y hace sol, en Python se escribiría: P and Q



Identifica todas las proposiciones en las siguientes oraciones y abrévielas con símbolos tales como P, Q o R, entonces convierta las oraciones al cálculo proposicional:

1) El coche que escapó era rojo o marrón

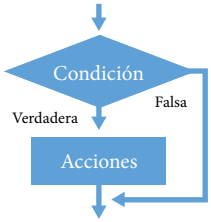
2) Si Jaime está en el granero, entonces Javier debe estar en el granero también

3) Las noticias no son buenas

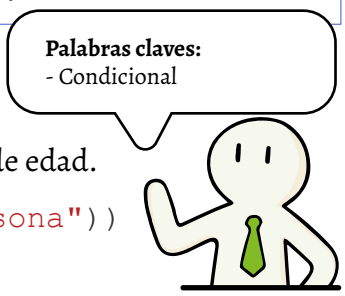
4) Llegarás a tiempo solo si te das prisa

5) Si ella estaba allí, entonces debió haber visto

Ejercicio 3. Estructura selectiva



Se utilizan para tomar decisiones lógicas, las estructuras selectivas pueden ser: simples, dobles o múltiples. La representación de una estructura selectiva se hace con palabras en Pseudocódigo (Si, entonces) (Joyanes, 2008) o bien en el lenguaje Python (if, else, elif).



Ejemplo: Elabora un programa que dada la edad de una persona, diga si es mayor o menor de edad y si no que diga que es menor de edad.

```
edad = int(input("Ingrese la edad de la persona"))
if (edad >= 18):
    print ("Es mayor de edad")
else:
    print ("Es menor de edad")
```

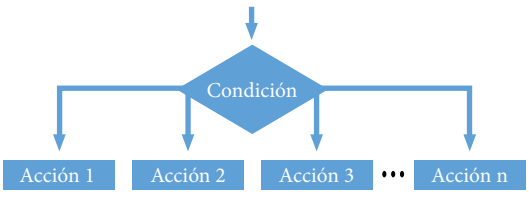
Realice estos programas con lápiz y papel sin usar el computador.

1) Elabore un programa que lea un número y calcule e imprima su cuadrado.

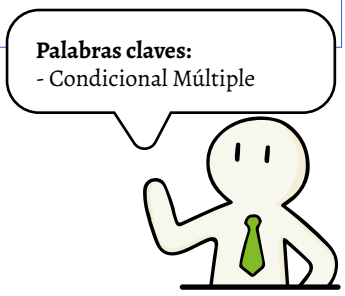
2) Elabore un programa que intercambie el valor de dos variables.

3) Elabore un programa que lea dos números e imprima la suma, resta, multiplicación y división.

Ejercicio 4. Estructura selectiva múltiple



Permite seleccionar, por medio de una expresión, el siguiente bloque de instrucciones a ejecutar de entre varios posibles.



Ejemplo: Elabore un programa en Python que dado un numero de la semana escriba que día pertenece. Ejemplo, Si es **1** escriba "**LUNES**", si es **2** que escriba "**MARTES**" y asi sucesivamente.

```
dia = int(input("Ingrese dia: "))
if (dia == 1):
```

```
    print ("LUNES")
elif (dia == 2):
    print ("MARTES")
elif (dia == 3):
    print ("MIERCOLES")
elif (dia == 4):
    print ("JUEVES")
elif (dia == 5):
    print ("VIERNES")
elif (dia == 6):
    print ("SABADO")
elif (dia == 7):
    print ("DOMINGO")
else:
    print ("NUMERO INVALIDO")
```

Realice estos programas con lápiz y papel sin usar el computador.

1) Elaborar un algoritmo que lea un numero entre 1 y 3 y escriba su valor en letras. Ejemplo 1 es igual a "Uno".

4. Estructura Repetitiva while – for

4.1. Laboratorio validación de un número

Objetivos de la actividad

- Resolver el caso con estructuras de datos, técnicas iterativas, procesos y validación de datos.
- Diseñar correctamente una estructura repetitiva que permita validar un dato de entrada de tipo numérico.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Saber gestionar el flujo de los datos de acuerdo con la planeación de la solución.
- Ser creativo al implementar las instrucciones de control.

Enunciado

Diseñe un algoritmo que valide una variable de tipo numérico (**int**) cuyo dato se ingresa por teclado, y en caso de que sea exitosa la validación, se debe mostrar un indicador numérico, el cual será el mismo número ingresado y en caso contrario, es decir, que la validación no haya sido exitosa se debe repetir nuevamente la solicitud del dato de entrada.

La condición para que la validación sea exitosa es que la variable de tipo entero **(int)** sea superior a **1**.

Ejemplo, si se ingresa mediante teclado el número **-1**, la validación será errónea, por lo tanto, se debe volver a solicitar el número. Y, si el número ingresado es superior a **1**, por ejemplo, el número **10**, el dato de salida será el mismo número ingresado, es decir el número **10**.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
numero	Numérico entero (int)	Número para validar.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
numero	Numérico entero (int)	Si la variable está validada correctamente se debe mostrar el mismo número ingresado . En caso contrario, se debe solicitar el número nuevamente, hasta que sea exitosa la validación.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
Ingrese un número superior a 1: 2
2
```

Prueba 2:

```
Ingrese un número superior a 1: -1
Error... Ingrese un número superior a 1: 10
10
```

Se recomienda que la función cread por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def validaNumero():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
validaNumero()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

4.2. Laboratorio validación de un texto

Objetivos de la actividad

- Diseñar correctamente una estructura repetitiva que permita validar un dato de entrada tipo texto.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Saber gestionar el flujo de los datos, de acuerdo con la planeación de la solución.
- Ser creativo al implementar las instrucciones de control.

Enunciado

Diseñe un algoritmo que valide una variable de tipo texto (**str**), cuyo dato se ingresa por teclado, y en caso de que sea exitosa la validación se debe mostrar el mensaje "**ok**". En caso contrario, es decir que la validación no haya sido exitosa se debe repetir nuevamente la solicitud del dato de entrada.

La condición para que la validación sea exitosa es que la variable de tipo texto **(int)** sea alguno de los siguientes textos: "**si**", "**Si**", "**sI**", "**SI**", "**no**", "**No**", "**nO**" o "**NO**".

Ejemplo:

Si se ingresa mediante teclado el texto "**sip**", la validación será errónea, por lo tanto, se debe volver a solicitar el texto. Y, si el texto ingresado es igual a alguno de los textos válidos, por ejemplo, el texto "**Si**", el dato de salida será el mensaje "**ok**".

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
x	Cadena de texto (str)	Texto para validar.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
mensaje	Cadena de texto (str)	Si la variable está validada correctamente se debe mostrar el mensaje "**ok**". En caso contrario, se debe solicitar el texto nuevamente hasta que sea exitosa la validación.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
si o no: si
ok
```

Prueba 2:

```
si o no: sip
Error...si o no: Si
ok
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def validaTexto():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
validaTexto()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

4.3. Laboratorio productoria

Objetivos de la actividad

- Diseñar correctamente una estructura repetitiva que permita resolver una productoria.
- Validar correctamente un dato de entrada numérico.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Reconocer los conceptos asociados a variables acumuladoras y su uso.
- Apropiar las diferentes técnicas iterativas.

Enunciado

Una productoria se simboliza con la letra griega pi mayúscula **Π** , la cual se lee *producto de*. Esta consiste en una multiplicación sucesiva de **n** términos.

Diseñe un programa que resuelva la siguiente productoria:

$$\prod_{i=1}^{n}(3i-2)$$

Donde,

n, corresponde al límite superior de la productoria

Π, es el símbolo que representa la productoria

i, es el elemento genérico de la productoria

i = 1, es el límite inferior de la productoria

Tenga en cuenta que el límite superior **n** de la productoria debe ser superior o igual a **1**, esto significa que hay que validar el valor de **n**. En caso de que **n** sea inferior a **1** el resultado debe ser **0** y se debe volver a solicitar el valor de **n** hasta que la validación sea exitosa.

Ejemplo: Si *n* = 0, entonces, *resultado* = 0

En este caso como **n** es igual a **0**, el resultado es **0**, y se debe volver a solicitar el valor de **n**.

Si *n* = 2, entonces, *resultado* = ((3 * 1) - 2) * ((3 * 2) - 2) = 1 * 4 = 4

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
n	Numérico entero (int)	Límite superior de la productoria.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
resultado	Numérico entero (int)	Es **0** si el límite superior **n** de la productoria es inferior a **1**, esto significa que hay que validar el valor de **n**. Se debe solicitar nuevamente el valor de **n** hasta que sea superior a **1**. En caso de que **n** sea superior o igual a **1**, el resultado debe ser la productoria realizada.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
Ingrese n: -1
0
Ingrese n nuevamente: 0
0
Ingrese n nuevamente: 1
1
```

Prueba 2:

```
Ingrese n: 4
280
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def productoria():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
productoria()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

4.4. Laboratorio divisor de un número

Objetivos de la actividad

- Usar del operador módulo de Python para el cálculo del resto de la división entre dos números.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Reconocer los conceptos asociados a los operadores aritméticos.
- Ser innovador al seleccionar las instrucciones repetitivas.

Enunciado

Los divisores de un número entero (int) son todos los números que al dividirlos su residuo es exactamente cero. Por ejemplo, los divisores del número 20 son: 1, 2, 4, 5, 10, 20.

Teniendo en cuenta que el operador % calcula el resto de la división entre dos números diseñe un algoritmo que dado un número entero (**int**) superior a **0** ingresado por teclado, muestre como salida los divisores de dicho número. Si el número es inferior o igual a **0** la salida debe ser exactamente el número **0**.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
numero	Numérico entero (int)	Límite superior de la productoria.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
divisor	Numérico entero (int)	Es **0** si el número ingresado es inferior a **1.** De lo contrario, se calculan los divisores del número y se imprimen uno a uno sin formatear la salida.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
Ingrese un numero: 2
1
2
```

Prueba 2:

```
Ingrese un numero: -1
0
```

Prueba 3:

```
Ingrese un numero: 20
1
2
4
5
10
20
```

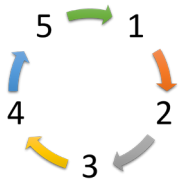
Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def divisores():

    # Desarrolle aquí su algoritmo

    pass
# Llamado de la función
divisores()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

4.5. Ejercicios de autoevaluación de Estructura Repetitiva



Una estructura repetitiva es una instrucción que hace que se repitan un conjunto de instrucciones, ya sean básicas, de control o complementarias (Joyanes, 2008).

Palabras claves:
- Estructura repetitiva
- Prueba de escritorio

Prueba de escritorio: Consiste en asignar datos de prueba o datos especificados por el programa para comprobar que la ejecución y la salida sean exitosas. Se puede representar mediante una tabla, dividida en filas y columnas, donde en cada columna se escribe una variable y en cada fila se van asignando los valores correspondientes, de acuerdo con el algoritmo, esto permite hacer una representación gráfica de la memoria del pc. (Cairó, 2005).

Ejemplo 1: Elabore un programa que imprima los números del 1 al 3 y muestre su prueba de escritorio.

```
i = 1
while (i <= 3):
    print (i)
    i = i + 1
```

iteración	i	i <= 3	Salida (print)
1	1	¿1 <= 3? = True	1
2	2	¿2 <= 3? = True	2
3	3	¿3 <= 3? = True	3
4	4	¿4 <= 3? = False	-

Ejemplo 2: El factorial de un número es una multiplicación sucesiva desde 1 hasta el número. El factorial de 0 es 1, y el factorial de 1 es 1. Escribe un programa en Python que calcule el factorial de un número.

```
n = int(input ("Ingrese un numero: "))
factorial = 1
```

```
i = 1
while (i<=n):
    factorial = factorial * i
    i = i + 1
print ("El factorial es: ", factorial)
```

n = 3				
iteración	**i**	**i <= 3**	**factorial**	**Salida (print)**
1	1	¿1 <= 3? = True	1 * 1 = 1	-
2	2	¿2 <= 3? = True	1 * 2 = 2	-
3	3	¿3 <= 3? = True	2 * 3 = 6	-
4	4	¿4 <= 3? = False	-	6

Ejercicio 1. Ciclo repetitivo while

Elabore un programa que lea n números e imprima teniendo en cuenta las siguientes consideraciones:

a. Cantidad de números positivos

b. Suma total de los números leídos

c. Promedio de los números positivos

d. Porcentaje de los ceros

Ejercicio 2. Prueba de escritorio

Realice la prueba de escritorio del siguiente segmento de código Python y muestre la salida paso por paso.

```
def ejercicio():
    n = 5
    i = 1
    sum = 0
    while ( i <= n ):
        sum = sum + i
        i = i + 1
    print ("La sumatoria de los números del 1 al 5 es: ", sum)
ejercicio ()
```

5. Estructura de datos – Fundamentos

5.1. Laboratorio cálculo de la media aritmética

Objetivos de la actividad

- Implementar y utilizar correctamente la estructura de una lista de Python.
- Diseñar correctamente una estructura de datos que permita resolver el promedio aritmético de una lista, mediante técnicas iterativas, procesos y validación de datos.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser organizado al analizar y resolver el problema.
- Reconocer las técnicas de manipulación de estructuras de datos.

Enunciado

Para un conjunto dado de números $x_1 x_2, \ldots, x_n$ la medida más conocida y útil del centro es la media o promedio aritmético del conjunto. Como casi siempre se pensará que los números x_i constituyen una muestra, a menudo se hará referencia al promedio aritmético como la media muestral.

La media muestral de las observaciones está dada por:

$$\bar{x} = \frac{x_1 + x_2 +, \ldots, + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Donde, el numerador de $\bar{x}$ se escribe más informalmente como $\sum x_i$, donde la suma incluye todas las observaciones muestrales, y n es la cantidad de valores.

Diseñe un algoritmo que dado como dato de entrada un conjunto de n números reales (**float**) almacenados en una lista, calcule el promedio aritmético.

Tenga en cuenta que la cantidad de valores **n** debe ser superior o igual a **1**. En caso contrario, se debe solicitar nuevamente el valor de la variable **n**.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
n	Numérico entero (int)	Cantidad de valores de la lista.
xi	Numérico real (float)	Números reales (float) ingresados uno a uno en una lista de **n** cantidad de valores, para realizar el cálculo de la media aritmética.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
promedio	Numérico real (float)	Corresponde al resultado de la media aritmética calculada.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
¿Cuántos datos son? 0
Error... ¿Cuántos datos son? 3
Por favor ingrese un número: 20.5
Por favor ingrese un número: 30.154
Por favor ingrese un número: 50.4124
33.69
```

Prueba 2:

```
¿Cuántos datos son? -1
Error... ¿Cuántos datos son? 0
```

```
Error... ¿Cuántos datos son? -100
Error... ¿Cuántos datos son? 1
Por favor ingrese un número: 50.6524
50.65
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def listaProm ():
    # Desarrolle aquí su algoritmo
    Pass
# Llamado de la función
listaProm ()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

5.2. Laboratorio porcentaje

Objetivos de la actividad

- Implementar y utilizar correctamente la estructura de una lista de Python.
- Diseñar correctamente una estructura de datos que permita resolver porcentajes de una lista, mediante técnicas iterativas, procesos y validación de datos.
- Utilizar las instrucciones de decisión, con condiciones simples y compuestas para la solución de un problema, de manera que sea posible considerar distintas opciones de solución.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Reconocer las técnicas de manipulación de estructuras de datos.
- Ser creativo al realizar la abstracción para la selección de los tipos de datos correctos.
- Comprender las diferentes salidas del problema propuesto.

Enunciado

Se quiere obtener la estadística de los pesos de los alumnos de un curso de n estudiantes de acuerdo a la siguiente información:

1. Porcentaje de alumnos con peso menor que 40 Kg
2. Porcentaje de alumnos con peso mayor o igual a 40 Kg y menor que 50 Kg
3. Porcentaje de alumnos con peso mayor o igual a 50 Kg y menor o igual que 60 Kg
4. Porcentaje de alumnos con peso mayor a 60 Kg

Tenga en cuenta que la cantidad de valores n debe ser superior o igual a 1.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
n	Numérico entero (int)	Cantidad de valores de la lista.
x	Numérico real (float)	Pesos reales (float) ingresados uno a uno en una lista de n cantidad de valores, para realizar el cálculo de los porcentajes.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
porcentaje1	Numérico real (float)	Corresponde al resultado de los alumnos con peso menor que 40.
porcentaje2	Numérico real (float)	Corresponde al resultado de los alumnos con peso mayor o igual a 40 Kg y menor que 50 Kg.
porcentaje3	Numérico real (float)	Corresponde al resultado de los alumnos con peso mayor o igual a 50 Kg y menor o igual que 60 Kg.
porcentaje4	Numérico real (float)	Corresponde al resultado de los alumnos con peso mayor a 60 Kg.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Nota: En este ejercicio se tienen en cuenta todas las cifras decimales.

Prueba 1:

```
¿Cuántos datos son? 0
¿Cuántos datos son? 10

Ingrese un peso: 45
Ingrese un peso: 50
Ingrese un peso: 51
Ingrese un peso: 42
Ingrese un peso: 65
Ingrese un peso: 60
Ingrese un peso: 57
Ingrese un peso: 63
Ingrese un peso: 47
Ingrese un peso: 45

porcentaje1: 0
porcentaje2: 40
porcentaje3: 40
porcentaje4: 20
```

Prueba 2:

```
¿Cuántos datos son? -1
¿Cuántos datos son? 0
¿Cuántos datos son? -100
¿Cuántos datos son? 3
Ingrese un peso: 35
Ingrese un peso: 35
Ingrese un peso: 35

porcentaje1: 100.0
porcentaje2: 0.0
porcentaje3: 0.0
porcentaje4: 0.0
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def listaPorcentaje ():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
listaPorcentaje()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

5.3. Laboratorio porcentajes jubilación

Objetivos de la actividad

- Implementar y utilizar correctamente la estructura para listas relacionadas de Python.
- Diseñar correctamente una estructura de datos que permita utilizar las instrucciones de decisión, con condiciones simples y compuestas para la solución de un problema, de manera que sea posible considerar distintas opciones para resolverlo.
- Utilizar adecuadamente la medida estadística de porcentaje.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser creativo al realizar la abstracción para la selección de los tipos de datos correctos.
- Comprender las diferentes salidas del problema propuesto.

Enunciado

El ISS (Instituto de Seguros Sociales) requiere clasificar a las n personas que se jubilar**á**n en el año 2021. Existen tres tipos de jubilaciones: por edad, por antigüedad joven y por antigüedad adulta. Se necesita conocer el porcentaje de cada tipo de jubilados que habrá durante este año.

Los requerimientos para jubilarse son los siguientes:

- Las personas adscritas a la jubilación por edad deben tener 60 años o más y una antigüedad en su empleo de menos de 25 años.
- Las personas adscritas a la jubilación por antigüedad joven deben tener menos de 60 años y una antigüedad en su empleo de 25 años o más.
- Las personas adscritas a la jubilación por antigüedad adulta deben tener 60 años o más y una antigüedad en su empleo de 25 años o más.

Diseñe un algoritmo que permita clasificar a las n personas que se jubilar**á**n.

Tenga en cuenta que la cantidad de valores n debe ser superior o igual a 1.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
n	Numérico entero (int)	Cantidad de valores de la lista.
edad	Numérico real (float)	Valores reales (float) ingresados uno a uno en una lista de n cantidad de edades, para implementar las condicionales.
tiempo	Numérico real (float)	Valores reales ingresados uno a uno en una lista de n cantidad de tiempos de antigüedad, para implementar las condicionales.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
jubilacionEdad	Numérico real (float)	Corresponde al porcentaje de personas que se jubilan por edad. Con todas las cifras decimales.
jubilacionJoven	Numérico real (float)	Corresponde al porcentaje de personas que se jubilan jóvenes. Con todas las cifras decimales.
jubilacionAdulta	Numérico real (float)	Corresponde al porcentaje de personas que se jubilan Adultas. Con todas las cifras decimales.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
¿Cuántos datos son? 0
¿Cuántos datos son? 10
Ingrese edad:60
Ingrese tiempo:14
Ingrese edad:58
```

```
Ingrese tiempo:25
Ingrese edad:65
Ingrese tiempo:30
Ingrese edad:63
Ingrese tiempo:25
Ingrese edad:57
Ingrese tiempo:25
Ingrese edad:63
Ingrese tiempo:30
Ingrese edad:68
Ingrese tiempo:26
Ingrese edad:57
Ingrese tiempo:25
Ingrese edad:62
Ingrese tiempo:26
Ingrese edad:59
Ingrese tiempo:25
porcentaje jubilación edad: 10.0
porcentaje jubilación joven: 40.0
porcentaje jubilación adulta: 50.0
```

Prueba 2:

```
¿Cuántos datos son?:-1
¿Cuántos datos son?:0
¿Cuántos datos son?:3
Ingrese edad:19
Ingrese tiempo:2
Ingrese edad:35
Ingrese tiempo:10
Ingrese edad:60
Ingrese tiempo:25
porcentaje jubilación edad: 0.0
porcentaje jubilación joven: 0.0
porcentaje jubilación adulta: 33.33333333333333
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def listaJubilación ():

    # Desarrolle aquí su algoritmo

    pass
# Llamado de la función
listaJubilación ()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

5.4. Laboratorio invertir y ordenamiento

Objetivos de la actividad

- Implementar y utilizar correctamente la estructura de una lista de Python.
- Diseñar correctamente una estructura de datos que permita mediante técnicas iterativas, procesos para movilizar los datos en una lista.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser creativo al realizar la abstracción para la selección de los tipos de datos correctos.
- Comprender las diferentes salidas del problema propuesto.

Enunciado

Diseñar un algoritmo que lea 10 números reales y los almacene en una lista.

1. Realice el procedimiento para invertir y guardar los datos en una nueva lista
2. Realice el procedimiento para ordenar la lista original ascendentemente.

A partir de estos procedimientos, imprima la **lista original**, la **lista invertida** y **la lista ordenada**.

Tenga en cuenta que la cantidad de valores n debe ser superior o igual a 1.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
titulo	Cadena de texto (str)	El título debe mostrarse al inicio del programa, Este dato solo es un mensaje (**print**) que indica el nombre del programa. El texto debe ser igual a "**INVERTIR Y ORDENAR**" Este mensaje debe ir antes de todo el programa.
n	Numérico entero (int)	Cantidad de valores de la lista.
x	Numérico real (float)	Valores reales (float) ingresados uno a uno en una lista de n cantidad de datos.

Descripción de los datos de salida en orden

Salida	Tipo de dato	Descripción
lista1	Numérico real (float)	Lista de valores reales ingresados uno a uno en la **lista1**, sin ordenar. Es decir, con los datos ingresados originalmente.
lista2	Numérico real (float)	Lista nueva de valores reales invertidos a partir de la **lista1**.
lista1	Numérico real (float)	Lista de valores reales ingresados uno a uno en una **lista1**, ordenados ascendentemente.

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
INVERTIR Y ORDENAR
¿Cuántos datos son?:-1
¿Cuántos datos son?:0
¿Cuántos datos son?:-100
¿Cuántos datos son?:10
Ingrese número:12
Ingrese número:15
Ingrese número:32
Ingrese número:24
Ingrese número:16
Ingrese número:11
Ingrese número:19
Ingrese número:66
Ingrese número:42
Ingrese número:35

[12.0, 15.0, 32.0, 24.0, 16.0, 11.0, 19.0, 66.0, 42.0, 35.0]
[35.0, 42.0, 66.0, 19.0, 11.0, 16.0, 24.0, 32.0, 15.0, 12.0]
[11.0, 12.0, 15.0, 16.0, 19.0, 24.0, 32.0, 35.0, 42.0, 66.0]
```

Prueba 2:

```
INVERTIR Y ORDENAR
```

```
¿Cuántos datos son?:3
Ingrese número:5
Ingrese número:9
Ingrese número:2

[5.0, 9.0, 2.0]
[2.0, 9.0, 5.0]
[2.0, 5.0, 9.0]
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def listaInvertirOrdenar ():

    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
listaInvertirOrdenar ()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

5.5. Laboratorio consultar en una lista

Objetivos de la actividad

- Implementar y utilizar correctamente la estructura de una lista de Python.
- Diseñar correctamente una estructura de datos que permita recorrer la lista para buscar información y a la vez contabilizar el número de veces que aparece.

Competencias para desarrollar o fortalecer

Durante la realización del laboratorio se busca que el estudiante afiance y desarrolle las siguientes competencias:

- Ser creativo para hacer modificaciones en una estructura de datos.

Enunciado

Dada una lista que contiene 10 números enteros (int) introducidos por teclado, se le debe pedir un número al usuario y mostrar por pantalla la cantidad de veces que se repite este número en la estructura.

Tenga en cuenta que la cantidad de valores n debe ser superior o igual a 1.

Penalizaciones

Se miden mediante el porcentaje de similaridad de los ejercicios realizados, es decir, si el ejercicio que ha desarrollado en el laboratorio tiene un porcentaje de similaridad superior a 70% se debe someter a revisión por parte del docente y en caso de que el porcentaje de similaridad sea de 100% será anulado con nota de 0.0.

Descripción de los datos de entrada

Nombre	Tipo de dato	Descripción
titulo	Cadena de texto (str)	El título debe mostrarse al inicio del programa, Este dato solo es un mensaje (**print**) que indica el nombre del programa. El texto debe ser igual a "**CONSULTA**". Este mensaje debe ir antes de todo el programa.
n	Numérico entero (int)	Cantidad de valores de la lista.
x	Numérico entero (int)	Valores enteros (int) ingresados uno a uno en una lista de n cantidad de valores.
y	Número entero(int)	Consiste en el valor a consultar en la lista, el cual puede ser encontrado o no.

Descripción de los datos de salida

Salida	Tipo de dato	Descripción
y	Numérico entero (int)	Consiste en el valor a consultar en la lista, el cual puede ser encontrado o no.
cantidad	Numérico entero (int)	Corresponde a la cantidad de veces que el número consultado se repite.
mensaje2	Cadena de texto (str)	Si el valor a consultar no se encuentra en la lista se produce el siguiente mensaje "**No encontrado**". Si el valor a consultar se encuentra una vez el menaje será: "**el número fue encontrado 1 vez**". Si el valor a consultar se encuentra varias veces el mensaje será: "**el número fue encontrado 5 veces**".

Datos de prueba (Lo datos de entrada y salida son en el orden que se muestra)

Prueba 1:

```
CONSULTA
¿Cuántos datos son?:0
¿Cuántos datos son?:3
Ingrese número:1
Ingrese número:2
Ingrese número:3
Ingrese número a consultar:5
No encontrado
```

Prueba 2:

```
CONSULTA
¿Cuántos datos son?:-1
¿Cuántos datos son?:3
Ingrese número:1
Ingrese número:2
Ingrese número:3
Ingrese número a consultar:3
el número fue encontrado 1 vez
```

Se recomienda que la función creada por usted tenga la siguiente estructura:

```
# Nombre: Aquí escriba su nombre
# Fecha: Aquí escriba la fecha de creación
'''
Escriba aquí las conclusiones
'''
def listaConsulta ():
    # Desarrolle aquí su algoritmo
    pass
# Llamado de la función
listaConsulta ()
```

Conclusiones

Al inicio del programa realizado, en un bloque de comentarios exprese sus aprendizajes, dificultades y aspectos a mejorar teniendo en cuenta los objetivos del laboratorio propuesto.

Anexo

Las autoras de este material quisiéramos enfatizar que el origen de esta guía pedagógica procede de la experiencia acumulada de años de docencia universitaria en el área de la Ingeniería de Sistemas, en particular en la materia de primer año denominada Fundamentos de Programación en la UTB, *Universidad Tecnológica de Bolívar.* Esta guía usa el lenguaje de programación Python para practicar los conceptos aprendidos durante este específico curso. El laboratorio de trabajo en el cual el estudiante realizará las practicas se encuentra en SAVIO[1], facilitando así el acceso al entorno de trabajo para los estudiantes matriculados en la asignatura.

La herramienta de trabajo se denomina VPL, *Virtual Programming Lab*, y funciona como complemento para las prácticas independientes de ejercicios de programación, como indicó Rodríguez del Pino et al. (2012), la herramienta VPL permite mejorar las habilidades de programación, siendo en el sistema de aprendizaje una característica esencial para mejorar el desempeño del estudiante para realzar sus prácticas.

El uso de la herramienta VPL presenta ventajas en el curso de Fundamentos de Programación, puesto que se logra un mayor nivel de compromiso y se despierta un mayor nivel de motivación, lo que lleva a mayores tasas de ejecución y mejores resultados de aprendizaje. El trabajo colaborativo se estimula mediante la interacción libre, una característica que los estudiantes aprecian y el docente tiene información más completa para brindar retroalimentación temprana a los estudiantes (Serrano et al., 2021).

Finalmente, cada temática está soportada mediante materiales entregados, ya sea en clases presenciales, virtuales y/o a través de referencias bibliográficas suministradas previamente junto a la bibliografía complementaria con la que fue elaborado el programa de Ingeniería de Sistemas y Computación de la UTB.

Pasos para crear una práctica en el laboratorio virtual en SAVIO

Se presenta la creación de un laboratorio virtual de programación (VPL) por parte del estudiante en la plataforma de SAVIO con el fin de ejecutar programas realizados en el lenguaje de programación Python y se describe el paso a paso desde la creación hasta la ejecución y entrega de un ejercicio. Los pasos que se realizan a continuación son con un ejemplo introductorio:

1. Ingresar al curso matriculado en Savio y dirigirse a la semana donde se encuentra la

[1] SAVIO es una herramienta del modelo educativo de la Universidad Tecnológica de Bolívar basada en Moodle, que permite flexibilizar el proceso de enseñanza y aprendizaje en el sistema universitario, tanto presencial como virtualmente (Serrano y Narváez, 2010).

actividad de laboratorio. El ejemplo mostrado en la Figura 2 es un laboratorio virtual de programación que se encuentra en la semana 01.

Figura 6. Ejemplo de un laboratorio en la semana 01

2. Una vez se ingrese al laboratorio se mostrará la **Descripción** de la actividad, el enunciado de esta actividad puede estar en un documento compartido por el docente y se procederá a crear el laboratorio. Cada laboratorio tiene una fecha de inicio y una fecha límite de entrega. Esto se despliega en la descripción del curso, como se muestra en la Figura 3.

Figura 7. Descripción de un laboratorio

3. Se hace clic en la pestaña **Editar** y, a continuación, se escribe el nombre del archivo. En este caso se creará un archivo de tipo **Python** como se muestra en la Figura 4. En el nombre del fichero se escribe "**main.py**", y se debe poner la extensión "**.py**" en minúsculas. Por último se procede a hacer clic en el botón **ok**.

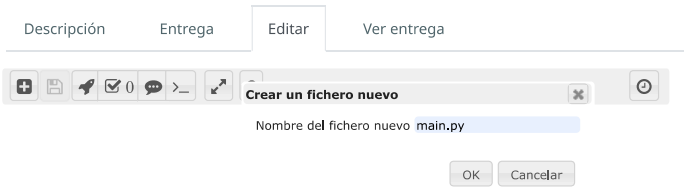


Figura 8. Creación de un algoritmo en Python en el laboratorio

4. Una vez creado el archivo "**main.py**" se procede a escribir el código en el laboratorio virtual. En el siguiente ejemplo Figura 5, se muestran algunas operaciones básicas realizadas en Python.

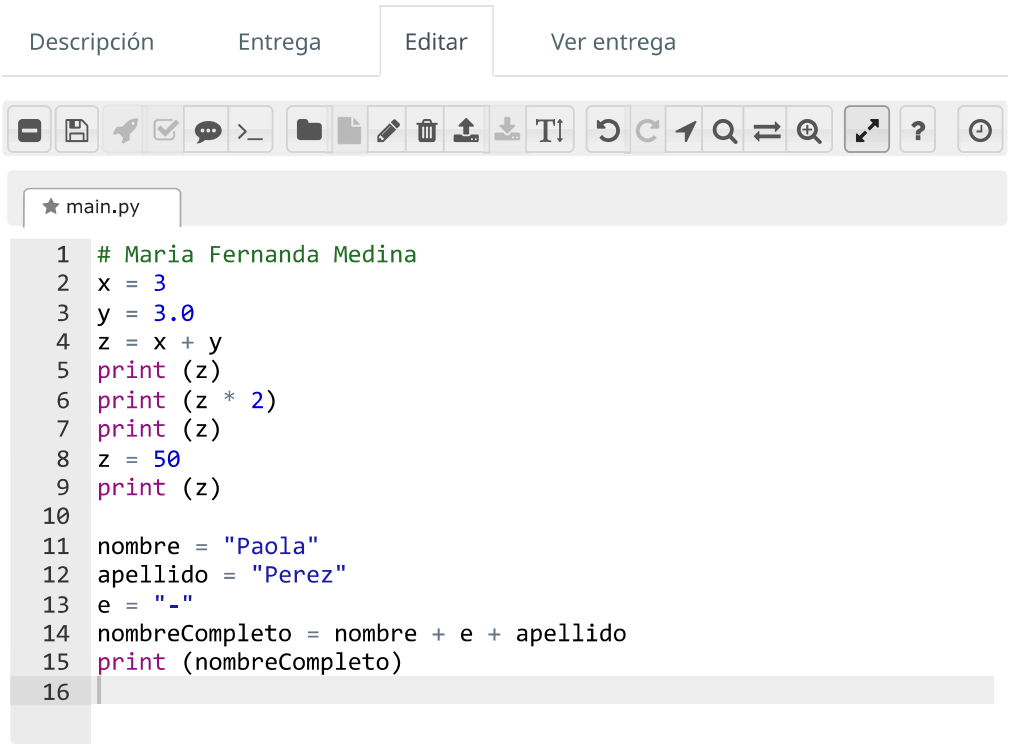


Figura 9. Operaciones básicas realizadas en Python, en el editor de VPL

5. Después que se escriba el código que se desea entregar se hace clic en el botón **Guardar** el cual tiene forma de diskette y luego se ejecuta haciendo clic el botón **Ejecutar** que tiene forma de cohete o presionando la combinación de teclas Ctrl + F11, mostrado en la Figura 6.

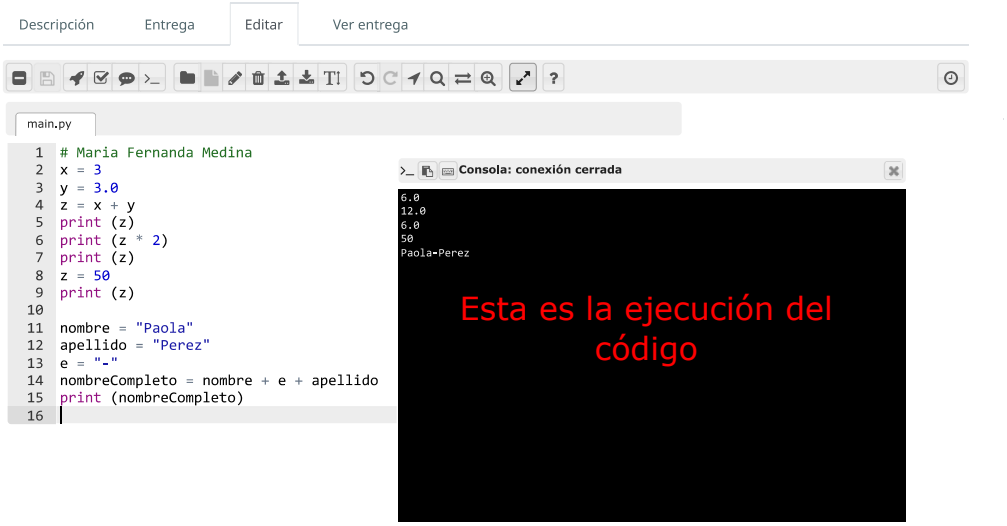


Figura 10. Ejecución de un programa de Python en el laboratorio

6. El trabajo se entrega automáticamente una vez se haga clic en guardar. Sin embargo, en caso de que desee comprobar que su entrega se ha realizado correctamente debe hacer clic en la pestaña **Ver entrega**, como se observa en la Figura 7, y puede observar que el trabajo fue entregado exitosamente.

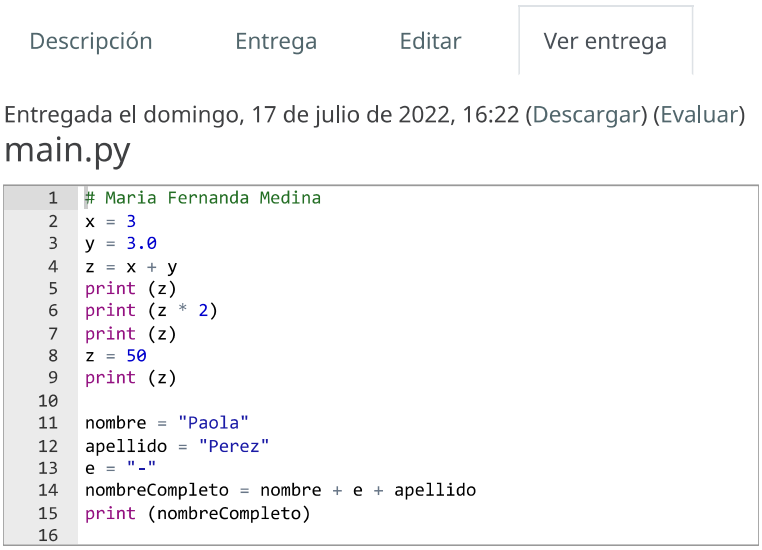


Figura 11. Entrega de un laboratorio de Python

Calificación automática de un trabajo enviado

En caso de que el ejercicio de la práctica de laboratorio lo permita y se desee visualizar una calificación previa del programa realizado, se debe hacer clic sobre el botón **Evaluar** o presionando la combinación de teclas Shift + F11, mostrado en la Figura 8, el cual se encuentra en la barra de herramientas. En la parte izquierda del editor se puede observar la *Nota propuesta*.

Figura 12. Evaluación de un ejercicio

En cada evaluación el programa arrojará una calificación, dependiendo del cumplimiento de los requerimientos, la calificación puede variar en un rango de 0.0 a 5.0.

***Nota**: Cada laboratorio se califica automáticamente, el docente decidirá si esta calificación se tendrá en cuenta o si solamente es para afianzar conocimientos del estudiante.*

Bibliografía

Cairó, O. (2005). *Metodología de la programación. Algoritmos, diagramas de flujo y programas*. Ciudad de México: Alfaomega.

Joyanes, L. (2008). *Fundamentos de programación. Algoritmos, estructura de datos y objetos*. Madrid: McGraw-Hill.

Liang, Y. D. (2013). *Introduction to Programming Using Python*. Upper Saddle River: Pearson.

Rodríguez del Pino, J. C., Rubio-Royo, E. y Hernández-Figueroa, Z. (2012). A virtual programming lab for Moodle with automatic assessment and anti-plagiarism features. En *International Conference on e-Learning, e-Business, Entreprise Information Systems, & e-Government* (pp. 1-6). Las Vegas: CSREA Press.

Serrano, J. E., Mantilla, J. C., Zúñiga, I., Henríquez, Y., Martínez-Santos, J. C. y Bautista, G. I. (2021). Desarrollo de Documentos Vivos y Experimentación con el Mundo Físico como Estrategias para la Retención y la Atracción de Estudiantes de Pregrado en los Cursos de Programación. En *19th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Prospective and trends in technology and skills for sustainable social development" "Leveraging emerging technologies to construct the future"*. Recuperado de: http://dx.doi.org/10.18687/LACCEI2021.1.1.381

Serrano, J. E. y Narváez, P. S. (2010). Uso de Software Libre para el Desarrollo de Contenidos Educativos. *Formación Universitaria*, *3*(6), 41-50. Recuperado de:

http://dx.doi.org/10.4067/S0718-50062010000600006

Material de consulta para mayor profundización

Deitel, P., y Deitel, H. (2022). *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud, Global Edition*. Harlow: Pearson.

Guttag, J. (2013). *Introduction to Computation and Programming Using Python*. Cambridge: Massachusetts Institute of Technology Press.

Python Software Foundation (Ed.) (2021). *Documentación de Python - 3.10.1. Recuperado de: https://docs.python.org/es/3/index.html*

Severance, C. (2015). *Python para informáticos: Explorando la información*. Ann Arbor: CreateSpace Independent Publishing Platform.

EDICIONES UTB

Universidad Tecnológica de Bolívar
CARTAGENA DE INDIAS