Actividades de Contextualización

**Retroalimentación**

* Completa el siguiente cuadro.

| Símbolos | Nombre | Función |
| --- | --- | --- |
|  | Inicio / Fin | Indica el comienzo o el fin de un proceso. |
|  | Línea de flujo | Muestra la secuencia de pasos o el flujo del proceso. |
|  | Entrada / Salida | Representa la entrada de datos al sistema o la salida de resultados. |
|  | Proceso / Actividad | Indica una operación, acción o instrucción a realizar. |
|  | Proceso / Actividad | Representa una decisión o condición que requiere una evaluación (Sí / No). |
|  | Conector interno | Utilizado para conectar diferentes partes del diagrama sin usar flechas largas. |
|  | Documento | Representa un documento generado o utilizado en el proceso. |

* Escribe correcto y/o incorrecto donde corresponda según las reglas de la construcción de algoritmos en diagramas de flujo y explica porque correcto o incorrecto.



1. **Inicio con símbolo ovalado (Proceso de inicio):**

Correcto. Se usa el símbolo correcto para el inicio.

1. **Paralelogramas usados para el "Proceso":**

Incorrecto: Los paralelogramos representan entradas o salidas de datos, no procesos. Aquí debieron usarse rectángulos.

1. **Conectores con flechas:**

Correcto. El flujo está bien indicado con flechas que van de arriba hacia abajo.

1. **Símbolo de fin (Fin Proceso):**

Correcto.: Se usa un óvalo, que es el símbolo adecuado para indicar el fin del diagrama.

* Escribe debajo de dado conjunto de flechas si es correcto o incorrecto la forma o posición de estas en un diagrama de flujo.



* Escribe si es correcto o incorrecto el siguiente diagrama, según la construcción de un diagrama de flujo.



* Escribe si es correcto o incorrecto el siguiente diagrama, según la construcción de un diagrama de flujo.



* Conceptualiza con tus palabras sin cortar y pegar de otros textos y responde en cada uno de los recuadros.

| ¿Qué es algoritmo?  Un algoritmo es un conjunto ordenado y finito de pasos o instrucciones que permiten resolver un problema o realizar una tarea específica. | ¿Qué es un diagrama de flujo?  Es una representación gráfica de un algoritmo o proceso, donde se utilizan símbolos para mostrar las operaciones, decisiones y el flujo de ejecución diagrama de flujo? |
| --- | --- |
| ¿Qué pasos son necesarios para solucionar un problema mediante un algoritmo?   1. Comprender y analizar el problema. 2. Identificar los datos de entrada y salida. 3. Diseñar el algoritmo con pasos lógicos. 4. Representarlo mediante pseudocódigo o diagrama de flujo. 5. Probarlo para verificar que soluciona el problema. 6. Corregir errores y optimizar si es necesario. | **¿Qué es pseudocódigo?**  El pseudocódigo es una descripción detallada de un algoritmo, escrita con una mezcla de lenguaje natural y elementos de programación, que permite entender la lógica sin usar un lenguaje de programación específico. |
| ¿Qué relación existe entre algoritmo, diagrama de flujo y pseudocódigo?  El algoritmo es la serie de pasos para resolver un problema.  El diagrama de flujo representa gráficamente ese algoritmo.  El pseudocódigo lo describe de forma textual y estructurada.  Todos buscan representar la lógica de una solución | **¿Qué diferencia existe entre diagrama de flujo y pseudocódigo?**  El diagrama de flujo utiliza símbolos y flechas para representar gráficamente el algoritmo.  El pseudocódigo lo expresa con palabras, de forma similar al lenguaje de programación pero sin sintaxis estricta. |
| ¿Cuál es la diferencia entre programa fuente y programa objeto?  El programa fuente es el código escrito por el programador en un lenguaje de alto nivel.  El programa objeto es el resultado de compilar ese código, ya traducido a lenguaje máquina. | **¿Cómo se analiza un problema para resolverlo a través de un algoritmo?**  Comprender el enunciado del problema.  Identificar qué se conoce (entradas) y qué se quiere obtener (salidas).  Dividir el problema en pasos lógicos y ordenados.  Diseñar una solución paso a paso.  Representarla como algoritmo. |
| ¿Cuáles son las características de un algoritmo?  Definido: cada paso debe ser claro.  Finito: debe terminar después de un número limitado de pasos.  Preciso: no debe haber ambigüedad.  Ordenado: sigue una secuencia lógica.  Eficiente: resuelve el problema con los recursos adecuados. | **Escribe las reglas para la elaboración de un diagrama de flujo.**  Iniciar con un símbolo de inicio y finalizar con uno de fin.  Usar los símbolos estándar correctamente.  Las flechas deben indicar el flujo del proceso.  El diagrama debe leerse de arriba hacia abajo o de izquierda a derecha.  No deben quedar líneas sin conectar.  Ser claro, limpio y fácil de entender. |
| ¿Qué es codificar un programa?  Codificar un programa es traducir un algoritmo a un lenguaje de programación específico para que pueda ser ejecutado por una computadora. | **¿Cuál es el objetivo de las pruebas tanto de escritorio como de computador y que las diferencia?**  Las **pruebas de escritorio** buscan simular manualmente el funcionamiento del algoritmo paso a paso para detectar errores lógicos.  Las **pruebas de computadora** ejecuta el programa para verificar su comportamiento real.  **Diferencia:** una es manual (escritorio), la otra automatizada (computadora). |
| ¿Cómo se representa un algoritmo mediante pseudocódigo?  Un algoritmo en pseudocódigo se representa usando instrucciones escritas en lenguaje natural estructurado, sin reglas estrictas de sintaxis, pero con lógica clara (por ejemplo: INICIO, LEER, SI, ENTONCES, FIN). | **Define variable y constante en un algoritmo computacional.**  **Variable:** espacio en memoria cuyo valor puede cambiar durante la ejecución del programa.  **Constante:** espacio en memoria con un valor fijo que no cambia durante la ejecución. |
| ¿Cómo se declara y se asigna una variable?  Para declarar una variable se indica su nombre y, a veces, su tipo de dato.  Para asignarla, se usa un operador como “←” o “=”.  Ejemplo en pseudocódigo: edad ← 20 | **¿Cuáles son las características de los nombres de variables y constantes?**  Deben ser **claros** y **descriptivos**.  No deben contener espacios.  Comienzan con una letra.  No deben ser palabras reservadas del lenguaje.  Las **constantes** suelen escribirse en mayúsculas para diferenciarlas. |

**Python**

Python es un lenguaje de programación que destaca por ser fácil de aprender y usar. Su diseño enfatiza la legibilidad del código, lo que lo convierte en una excelente opción para principiantes y expertos por igual.

**¿Por qué Python es popular en Inteligencia Artificial?**

1. **Facilidad de Uso**: Python tiene una sintaxis sencilla y clara, lo que permite a los desarrolladores concentrarse en resolver problemas en lugar de lidiar con la complejidad del código. Esto es especialmente útil en inteligencia artificial, donde el enfoque está en desarrollar y probar modelos complejos.
2. **Bibliotecas Especializadas**: Existen numerosas bibliotecas en Python diseñadas específicamente para tareas de inteligencia artificial y aprendizaje automático. Algunas de las más populares son:

**NumPy**: Para realizar operaciones matemáticas y manipulación de arrays.

**Pandas**: Para el manejo y análisis de grandes volúmenes de datos.

**TensorFlow** y **PyTorch**: Para crear y entrenar redes neuronales y modelos de deep learning.

**Scikit-learn**: Para implementar algoritmos de machine learning de manera sencilla.

1. **Comunidad y recursos**: Python tiene una comunidad enorme y activa, lo que significa que siempre hay soporte disponible, desde documentación detallada hasta foros y tutoriales. Esto facilita el aprendizaje y la resolución de problemas cuando se trabaja en proyectos de inteligencia artificial.
2. **Rápido Desarrollo**: La simplicidad de Python y la disponibilidad de herramientas listas para usar permiten que los desarrolladores experimenten y creen prototipos rápidamente, lo cual es crucial en el campo de la inteligencia artificial, donde es común probar diferentes enfoques para ver qué funciona mejor.
3. **Portabilidad y Versatilidad**: Python puede ejecutarse en diferentes sistemas operativos sin necesidad de modificar el código, lo que lo hace muy flexible. Además, se puede integrar fácilmente con otros lenguajes y tecnologías, lo que lo convierte en una opción versátil para proyectos de IA que requieren la combinación de diferentes herramientas.

En esta guía iniciaremos con un vistazo de algoritmos utilizando Python como lenguaje de programación:

**1. Introducción a Python**

**Conceptos Básicos:**

Sintaxis básica, variables, tipos de datos (enteros, flotantes, cadenas, listas, tuplas, diccionarios).

Operadores (aritméticos, de comparación, lógicos).

**Estructuras de Control:**

Condicionales (if, else, elif).

Bucles (for, while).

Comprensión de listas.

**Instalación**Un dibujo de una cara feliz

Descripción generada automáticamente con confianza media

<https://www.youtube.com/watch?v=EkZdeWBlqts>

**Conceptos Básicos:**Un dibujo de una cara feliz

Descripción generada automáticamente con confianza media

<https://www.youtube.com/watch?v=K-LTJKyffZQ>

***Comentarios***

Los comentarios en Python son líneas de texto que se incluyen en el código pero que no se ejecutan. Su propósito es ayudar a los desarrolladores a entender el código, explicar el propósito de ciertas secciones o dejar notas para futuros desarrolladores que trabajen en el mismo proyecto.

**Tipos de Comentarios en Python**

1. **Comentarios de una sola línea**: Se utilizan para añadir una breve descripción o explicación en una línea específica del código. En Python, se utilizan el símbolo # para indicar que lo que sigue en la línea es un comentario.

**Ejemplo**

1. **Comentarios multilínea**: Para escribir comentarios que abarcan varias líneas, se pueden utilizar múltiples símbolos # al comienzo de cada línea, o se pueden utilizar comillas triples (''' o """). Aunque el uso de comillas triples es más comúnmente asociado con cadenas de documentación (docstrings), a veces también se utilizan para comentarios multilínea.

**Ejemplo con #:**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Ejemplo con comillas triples:**

Texto

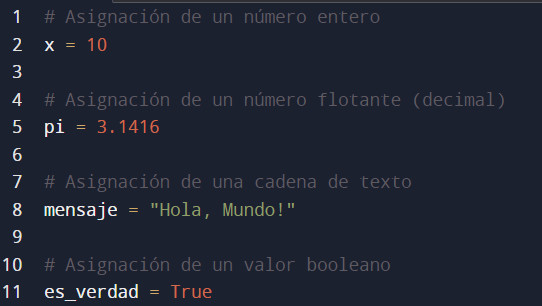
Descripción generada automáticamente

***Variables***

Las variables son un concepto fundamental en la programación, y en Python, juegan un papel clave en la construcción de algoritmos. Una variable es un nombre que se le da a un espacio en la memoria donde se almacena un valor. Este valor puede ser de diferentes tipos, como un número, una cadena de texto, una lista, entre otros.

**Declaración y Asignación de Variables en Python**

En Python, no es necesario declarar explícitamente el tipo de una variable. Simplemente se asigna un valor a una variable usando el operador igual (=).



Con el comando print(), puedo visualizar el contenido de las variables.

* Visualiza en consola el valor que almacenan las anteriores variables

**Nombres de Variables**

Los nombres de las variables deben seguir ciertas reglas:

* Deben comenzar con una letra (a-z, A-Z) o un guión bajo \_.
* Pueden contener letras, números (0-9) y guiones bajos.
* No pueden comenzar con un número.
* Son sensibles a mayúsculas y minúsculas, es decir, edad Edad son variables diferentes.
* No pueden tener en el nombre caracteres especiales (Solamente guión bajo)
* En la siguiente tabla encontrarás posibles identificadores de una variable, escribe correcto o incorrecto y explica por qué.

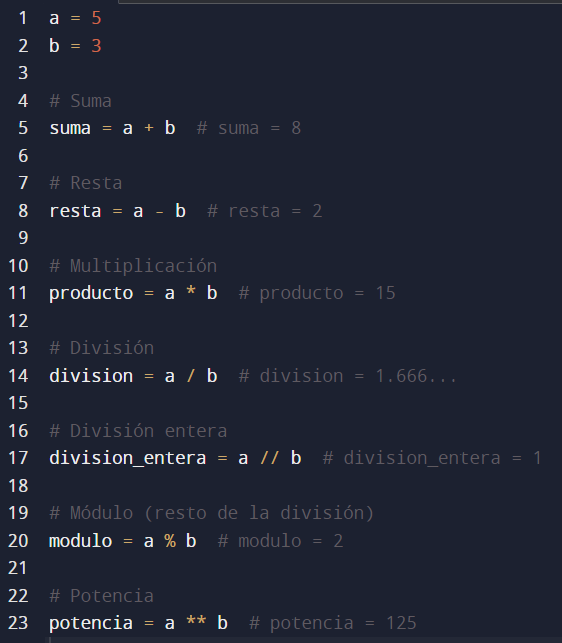
| Nombre | Correcto o Incorrecto | Justifica |
| --- | --- | --- |
| var1 | correcto | usa el número al final y junto |
| precio-2 | incorrecto | usa un guión |
| saldo\_2 | correcto | usa guión al piso |
| correo@gmail | incorrecto | usa el @ |
| 1\_variable | incorrecto | inicia con un número |
| 2-variable | incorrecto | inicia con número y usa guión |
| SaLdo\_FiNaL | incorrecto | mezcla mayúsculas con minúsculas |
| dos\*tres | incorrecto | usa ‘\*’ |
| 5+3 | incorrecto | usa ‘+’ |
| cincomastres | correcto | letras en minúsculas |

**Tipos de Datos Comunes en Variables**

* Enteros (int): Números sin parte decimal.
* Flotantes (float): Números con parte decimal.
* Cadenas (str): Texto.
* Booleanos (bool): Valores lógicos True o False.

**Operaciones con Variables**

Las variables pueden ser manipuladas mediante operaciones matemáticas o concatenación en el caso de cadenas.



* Considera que vari y var1 son variables de tipo entero.

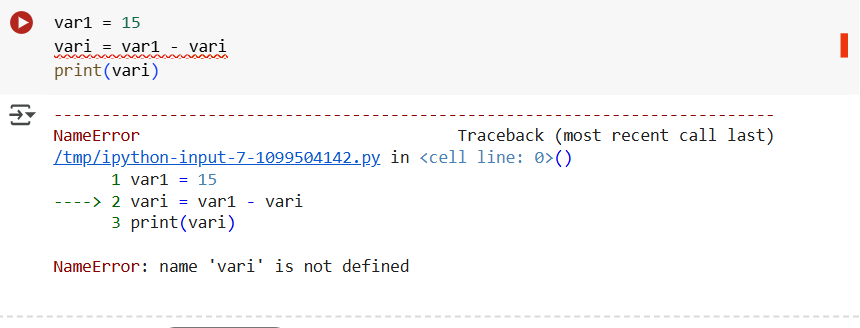
**Tarea**: Utiliza la plataforma [Programiz](https://www.programiz.com/python-programming/online-compiler/) para escribir las siguientes instrucciones en Python.

var1=15

**Resultado**: Ejecuta el código en [Programiz](https://www.programiz.com/python-programming/online-compiler/) y anota los resultados en la siguiente tabla.

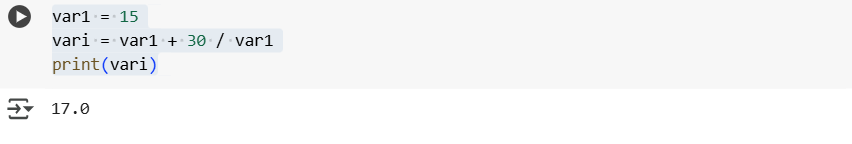
| Expresión | Valor en la variable vari | Explica |
| --- | --- | --- |
| vari = var1 - vari | error | La variable ‘vari’ no está definida |
| vari = var1 + 30 / var1 | 17.0 | Primero se hace la división 30 / 15 = 2.0, luego se suma 15 + 2.0 = 17.0 |
| vari = vari + 3 | error | La variable ‘vari’ no está definida |
| vari = var1 + 3 / 2 | 16.5 | Primero se divide 3 / 2 = 1.5 y luego se suma 15 + 1.5 = 16.5. |
| vari = var1 + 3 \* 2 | 21 | primero se multiplica 3 \* 2 = 6 y luego se suma 15 + 6 = 21. |
| vari = (var1 + 3) \* 2 | 36 | primero se suma 15 + 3 = 18 y luego se multiplica 18 \* 2 = 36. |

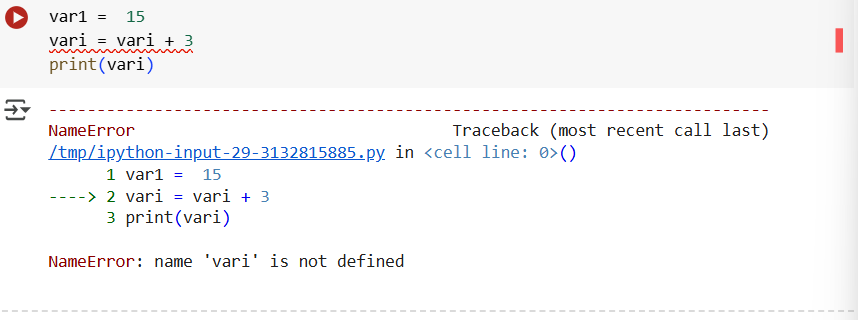
**SOLUCIÓN**

**1.**

**2.**

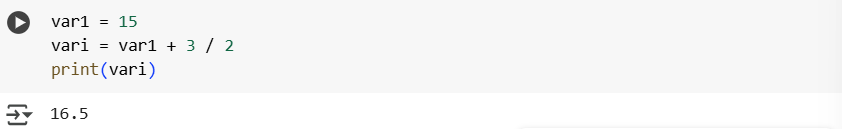
****

****

**3.**

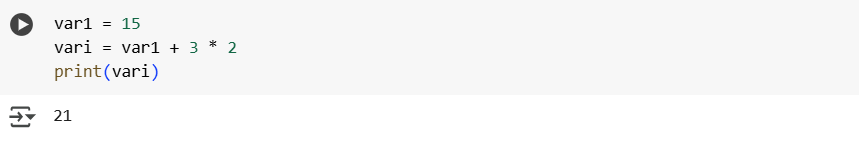
**4.**

****

****

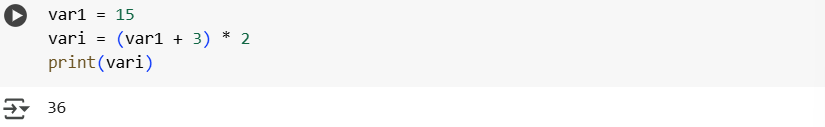
**5.**

****

****

**6.**

****

****

* Considerando que aux y aux1 son variables enteras.

**Tarea**: Utiliza la plataforma Programiz para escribir las siguientes instrucciones en Python.

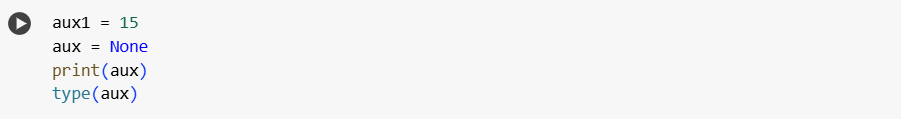
aux1=15

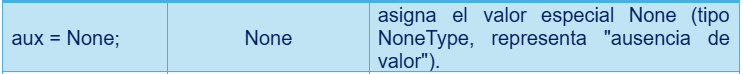
**Resultado**: Ejecuta el código en Programiz y anota los resultados en la siguiente tabla.

| Expresión | Valor en aux | Explica |
| --- | --- | --- |
| aux = None; | None | asigna el valor especial None (tipo NoneType, representa "ausencia de valor"). |
| aux = 90 / aux; | error | variable ‘aux’ no está definida |
| aux = 90 / aux1; | 6.0 | Se divide 90 entre 15 y se asigna el resultado (6.0) |
| aux = 0; | 0 | se le asigna en valor entero ‘0’ |
| aux = aux1 /aux; | error | ‘aux’ no está definido |

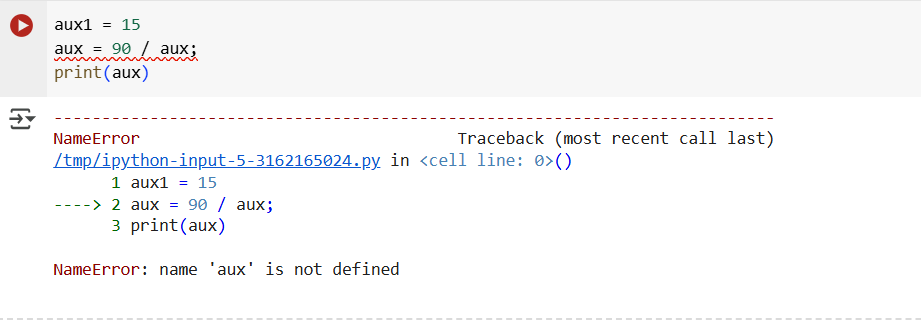
**SOLUCIÓN**

**1.**

****

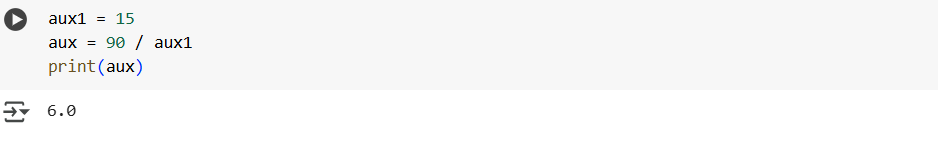
****

**2.**

****

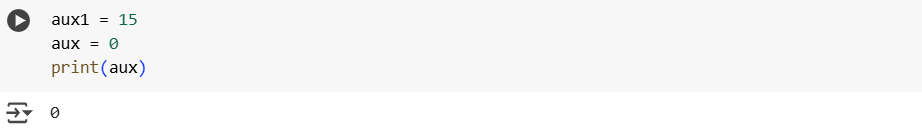
****

**3.**

****

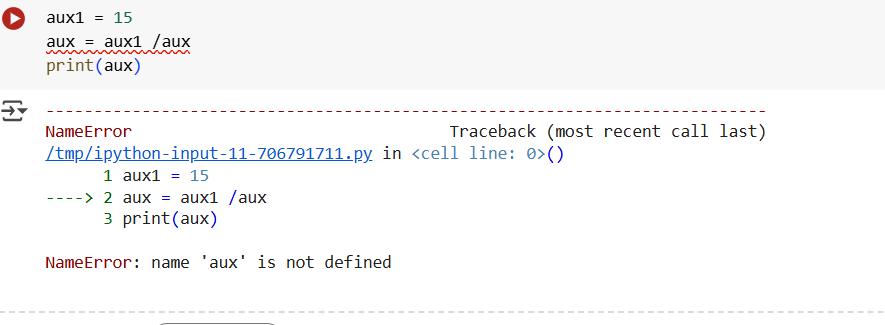
****

**4.**

****

****

**5.**

****

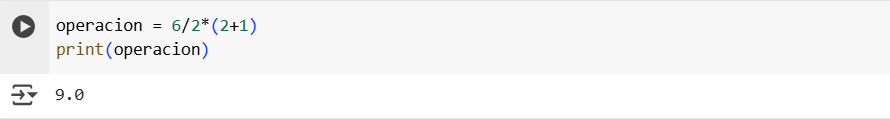
****

* En el cuadro a continuación, se presentan varias operaciones que almacenará un valor en una variable. En la columna titulada "Procedimiento Manual", escribe el resultado que obtienes al realizar la operación manualmente. Luego, utiliza el aplicativo Programiz para ejecutar las instrucciones en Python y anota el resultado en la columna "Programiz".

| Expresión | Procedimiento Manual | Programiz | Explica |
| --- | --- | --- | --- |
| operacion = 6/2\*(2+1) | 9 | 9 | jerarquía de los dignos |
| saldo = 45 + 18 / 9 | 47 | 47.0 | jerarquía de los dignos |
| saldo = (45 + 18) / 9 | 7 | 7.0 | jerarquía de los dignos |
| Operaciones2= 4+5\*30/3-10 | 44 | 44.0 | jerarquía de los dignos |
| aux = (15 - (8 - 5 )) · (5 + (6 - 4 )) - 3 + (8 - 6 ) | 83 | 83.0 | jerarquía de los dignos |
| Saldo\_3 = (100 + 45 / 5 \* 2) – 1 | 117 | 117.0 | jerarquía de los dignos |
| div = 34 / 11 | 3,09 | 3.090909090909 | jerarquía de los dignos |
| div\_2 = 34 //11 | 3 | 3.0 | división parte entera |
| par\_no\_par = 23 % 2 | 1 | 1.0 | residuo de la división |
| a = (42 + 20) ÷ 4 – 2 \* (9 ÷ 3) – 2 \* [18 + 3 \* (13 – 9) – 5] | -40.5 | -40.5 | jerarquía de los dignos |
| b={ [(7 + 5 \* 2) + 3 \* (3 \* 3) – ( 20 ÷ 5 ) ] } \* ( 9 – 2)} | 7.000 | 7000.0 | jerarquía de los dignos |
| C = 3 \* [ -3 + (- 3) ] – 14 ÷ (- 7) | 16 | 16.0 | jerarquía de los dignos |

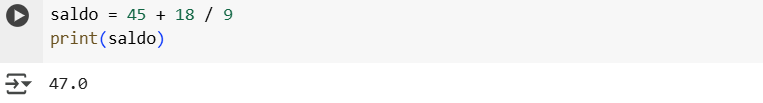
**SOLUCIÓN**

**1.**

****

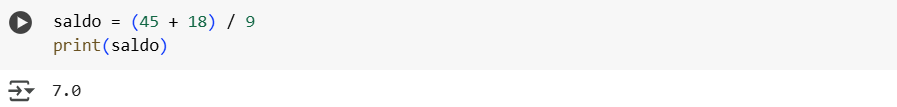
****

**2.**

****

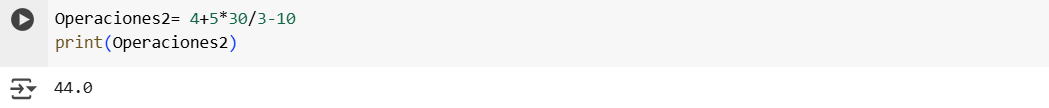
****

**3.**

****

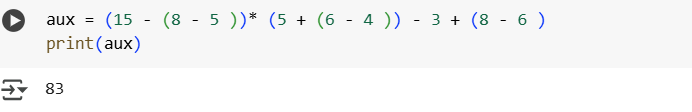
****

**4.**

****

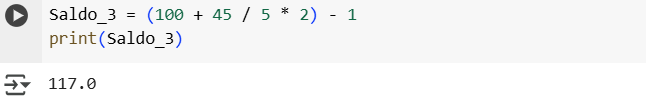
****

**5.**

****

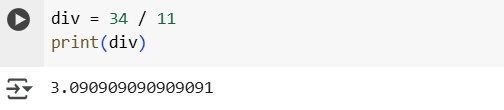
****

**6.**

****

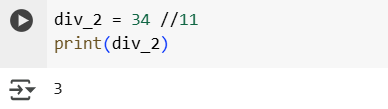
****

**7.**

****

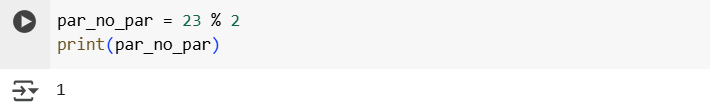
****

**8.**

****

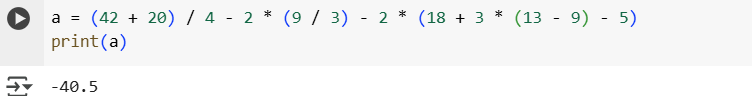
****

**9.**

****

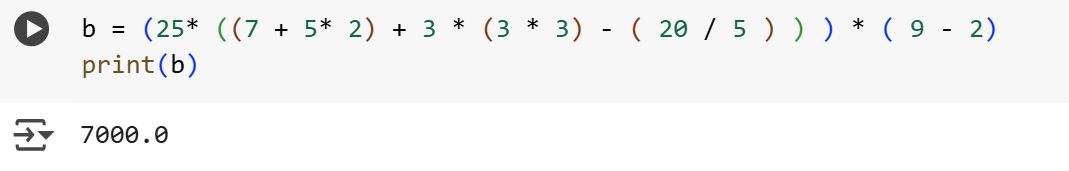
****

**10.**

****

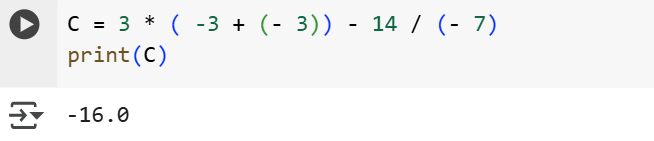
****

**11.**

****

****

**12.**

****

****

**Condicionales**

Los condicionales en programación son estructuras que permiten ejecutar diferentes bloques de código en función de ciertas condiciones. En Python, se utilizan para tomar decisiones y controlar el flujo del programa. Los condicionales permiten que el código reaccione de manera diferente según los valores de las variables o el estado del programa.

**Tablas de verdad**

* Observa el siguiente videoUn dibujo de una cara feliz

  Descripción generada automáticamente con confianza media

<https://www.youtube.com/watch?v=G53Da_gzsx0>

* Resuelve:

Ten en cuenta que ‘y’ se representa con &&, ‘o’ se representa con ||, ‘negación’ se representa con !

| p | q | r | p && q && r | p || q || r | p && q || r | ! (p && q || r) |
| --- | --- | --- | --- | --- | --- | --- |
| Falso | Falso | Falso | Falso | Falso | Falso | Verdadero |
| Falso | Falso | Verdadero | Falso | verdadero | verdadero | Falso |
| Falso | Verdadero | Falso | Falso | verdadero | Falso | Verdadero |
| Falso | Verdadero | Verdadero | Falso | verdadero | verdadero | Falso |
| Verdadero | Falso | Falso | Falso | verdadero | Falso | Verdadero |
| Verdadero | Falso | Verdadero | Falso | verdadero | verdadero | Falso |
| Verdadero | Verdadero | Falso | Falso | verdadero | verdadero | Falso |
| Verdadero | Verdadero | Verdadero | Verdadero | verdadero | verdadero | Falso |

* Resuelve:

| p | q | ¬p | ¬q | ¬p ∧ q | q ∨ ¬q | ¬p ∧ q ↔ q ∨ ¬q |
| --- | --- | --- | --- | --- | --- | --- |
| V | V | F | F | F | V | F |
| V | F | F | V | F | V | F |
| F | V | V | F | V | V | V |
| F | F | V | V | F | V | F |

* Analiza la siguiente tabla



* Contesta: ¿Qué es la tabla del código ASCII?

**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange (Código Estándar Americano para el Intercambio de Información)

El **código ASCII** (no *ASSCII*) es un **sistema de codificación de caracteres** que permite representar letras, números, signos de puntuación y algunos símbolos especiales en computadoras y dispositivos electrónicos.

* Teniendo en cuenta que las variables

i = 7 (entero)

f = 5.5 (real)

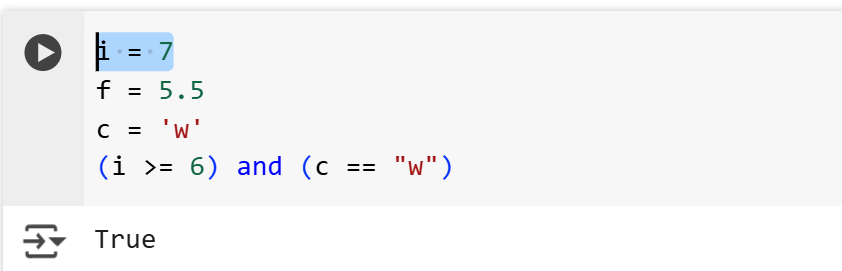
c = ‘w’ (caracter) (ASCII 119).

* Analiza la expresión completa de la siguiente tabla con su valor lógico y su valor binario (0 o 1).

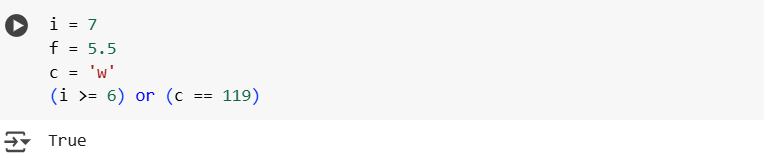
| expresión | Valor lógico | Valor binario |
| --- | --- | --- |
| (i >= 6) && (c == ‘w’) | Verdadero | 1 |
| (i >= 6) || (c == 119) | Verdadero | 1 |
| (f < 11) && (i > 100) | Falso | 0 |
| (c != ‘p’) || ((i+f) <= 10) | Verdadero | 1 |
| (i >= 6) || (c == 65) | Verdadero | 1 |
| (c != ‘p’) || ((i+f) >= 10)&& (c == ´W´) | Verdadero | 1 |
| (f < 11) || (i > 100) || (f=5.1) | Verdadero | 1 |
| ! ((i >= 6) && (c == ‘w’)) | Falso | 0 |

**SOLUCIÓN**

**1.**

****

**2.**



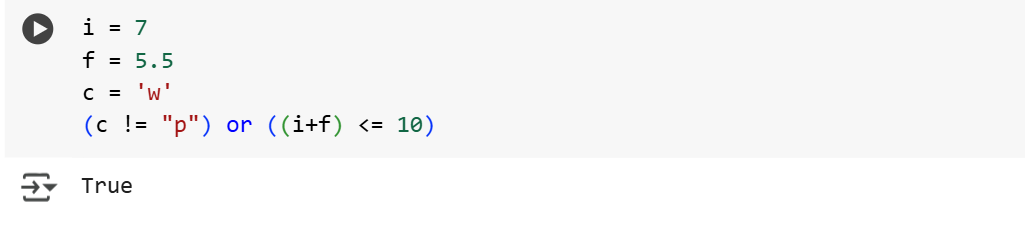


**3.**



****

**4.**



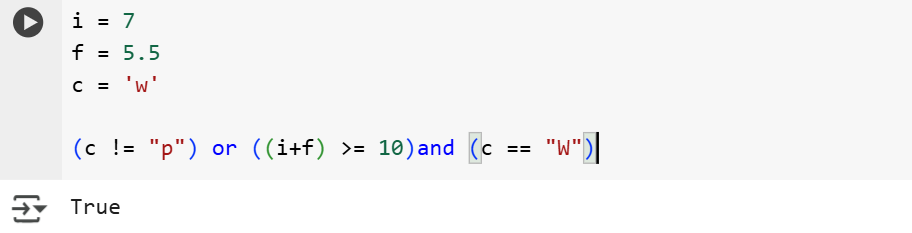


**5.**



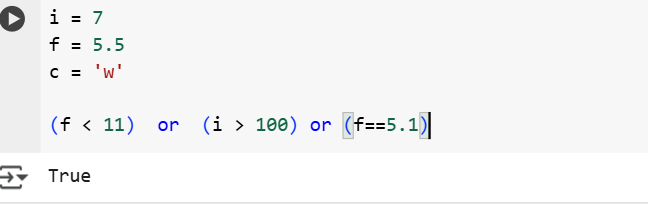


**6.**





**7.**





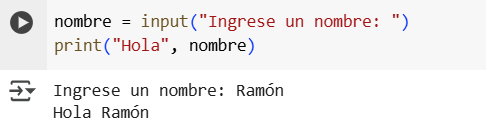
**8.**



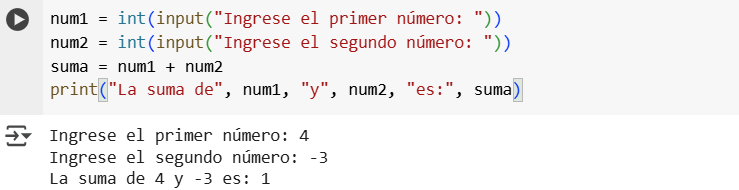


* Soluciones ejercicios utilizando Pseudocódigo en Js o Python.

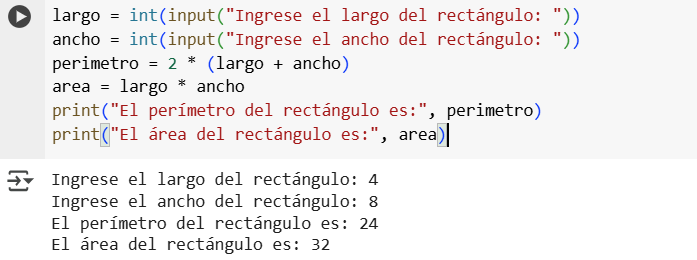
1. Codifique un algoritmo que solicite el nombre y devuelve como salida el mensaje: Hola nombre ingresado. Por ejemplo, si el usuario digita 'Ramón’, el mensaje desplegado será: ‘Hola Ramón’.



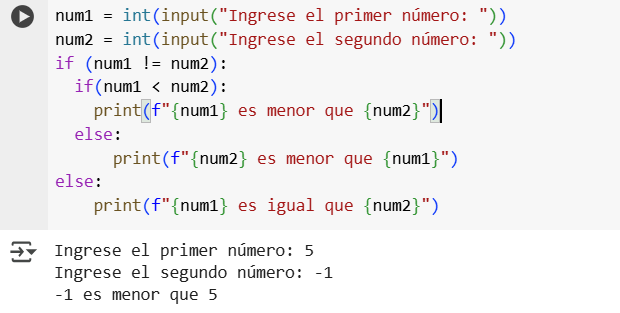
1. Realice un algoritmo que solicite dos números enteros, realice su suma y muestre el resultado.



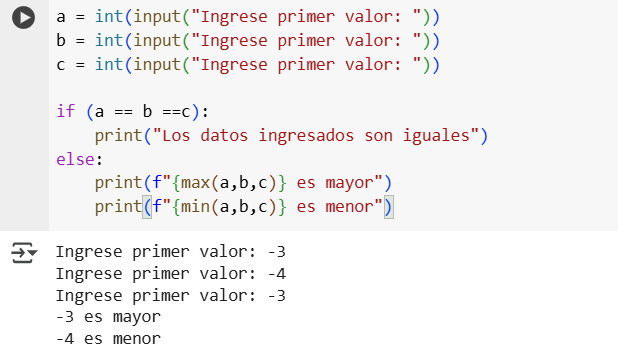
1. Realizar un algoritmo que calcule el perímetro y el área de un rectángulo dadas la base y la altura del mismo.



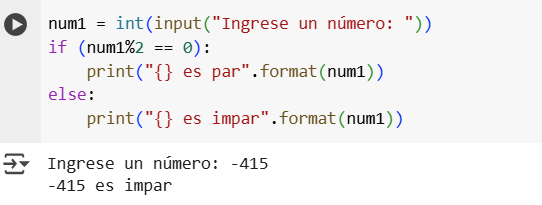
1. Desarrolle un algoritmo que permita leer dos valores distintos, determinar cuál de los dos valores es el mayor y escribirlo.



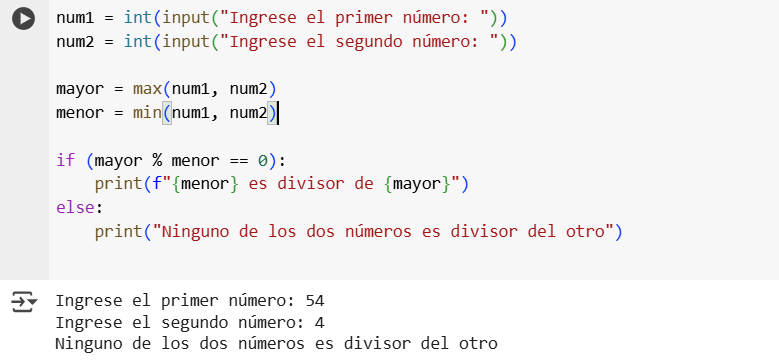
1. Desarrolle un algoritmo que permita leer tres valores y almacenarlos en las variables A, B y C respectivamente. El algoritmo debe imprimir cual es el mayor y cuál es el menor. Recuerde constatar que los tres valores introducidos por el teclado sean valores distintos. Presente un mensaje de alerta en caso de que se detecte la introducción de valores iguales.



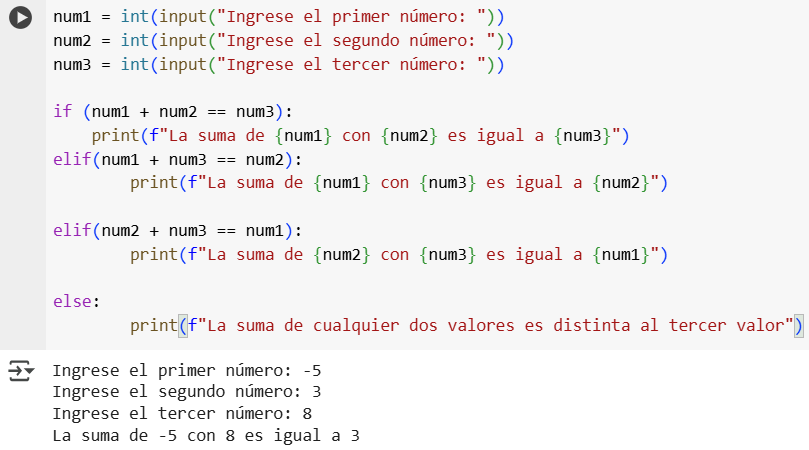
1. Dado un número imprimir un mensaje que diga si es par o impar.



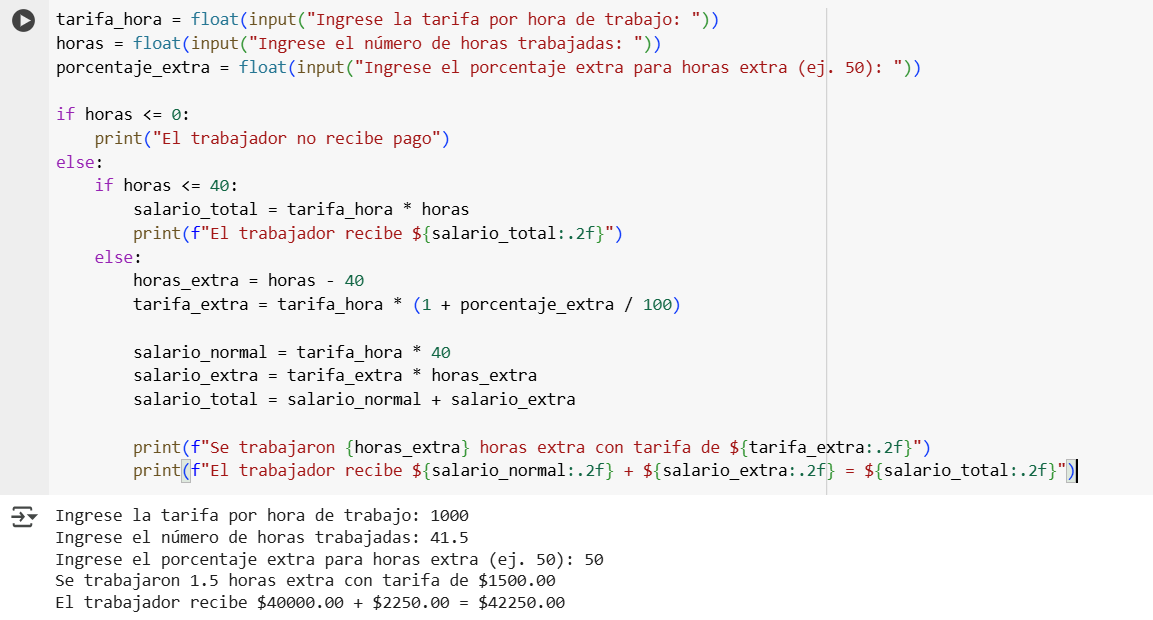
1. Averiguar si dados dos números leídos del teclado, uno es divisor de otro.



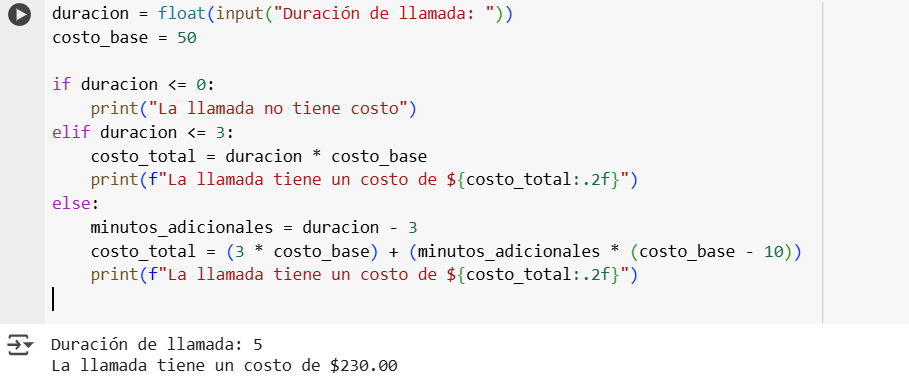
1. Leer tres números y determinar si la suma de cualquier pareja de ellos es igual al otro número. Si se cumple esta condición escribir "Iguales", en caso contrario, escribir "Distintas".



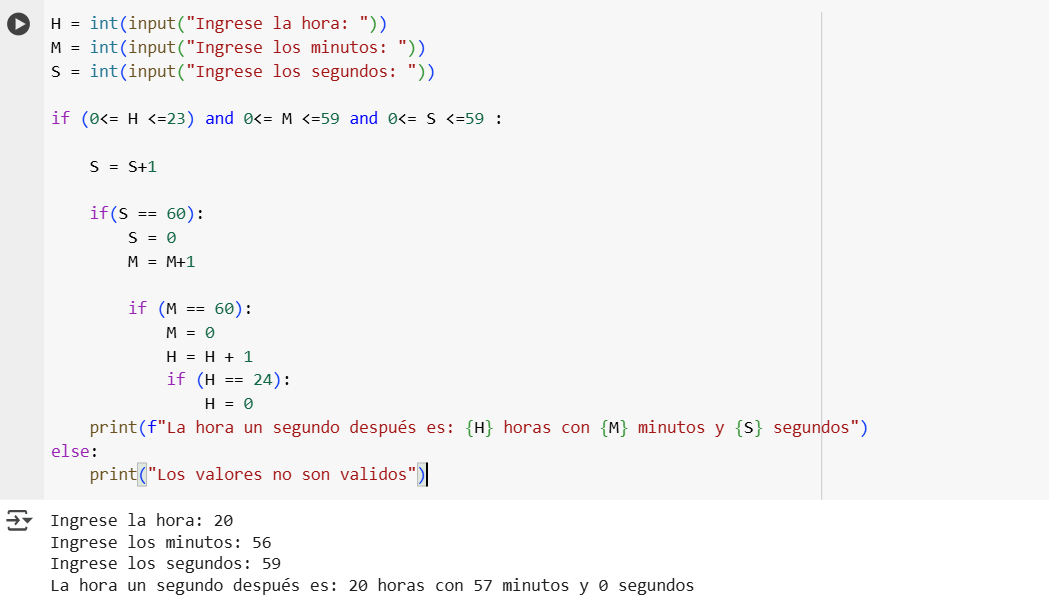
1. A un trabajador le pagan según sus horas y una tarifa de pago por horas. Si la cantidad de horas trabajadas es mayor a 40 horas. La tarifa se incrementa en un 50% para las horas extras. Calcular el salario del trabajador dadas las horas trabajadas y la tarifa.



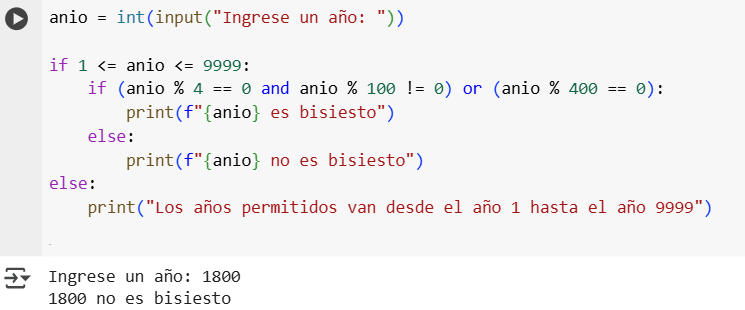
1. Dada la duración en minutos de una llamada calcular el costo, considerando: Hasta tres minutos el costo es $ 50 por minuto. Por encima de tres minutos es $ 50 menos $ 10 por cada minuto adicional.



1. Diseñar un algoritmo que me permita ingresar la hora, minutos y segundos y que me calcule la hora en el siguiente segundo ("0<= H <=23", "0<= M <=59" "0<= S<=59").



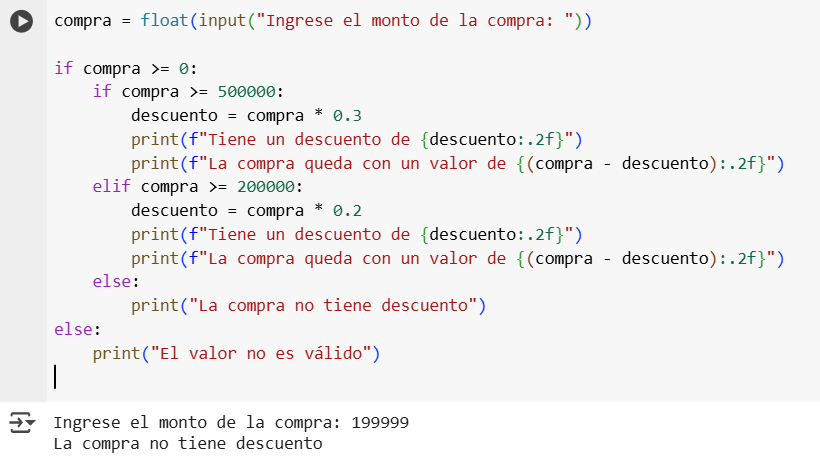
1. Escribir un algoritmo que determine si un año es bisiesto. Un año es bisiesto si es múltiplo de 4 (por ejemplo 1984). Los años múltiplos de 100 no son bisiestos, salvo si ellos son también múltiplos de 400 (2000 es bisiesto, pero; 1800 no lo es).



1. "ALMACENES BARATO BARATO" se encuentra de aniversario y ha programado una serie de ofertas con la finalidad de brindar facilidades a sus clientes y a la vez de incrementar sus ventas. Estas ofertas se basan específicamente en un porcentaje de descuento sobre el total de compra el cual varía de acuerdo al monto:

Por un monto mayor o igual a $500000 se hará un descuento del 30%.

Por un monto menor de $500000 pero mayor o igual a 200000 se hará un descuento del 20%



1. Katthy organiza una fiesta en la cual una computadora controla el ingreso mediante 5 claves. Si se ingresa al menos una clave incorrecta esta imprimirá "TE EQUIVOCASTE DE FIESTA" y no permitirá el ingreso. Si las 5 claves son correctas imprimirá "BIENVENIDO A LA FIESTA" Las Claves son:

1: "TIENES"

2: "QUE SER"

3: "INVITADO"

4: "PARA"

5: "INGRESAR"

