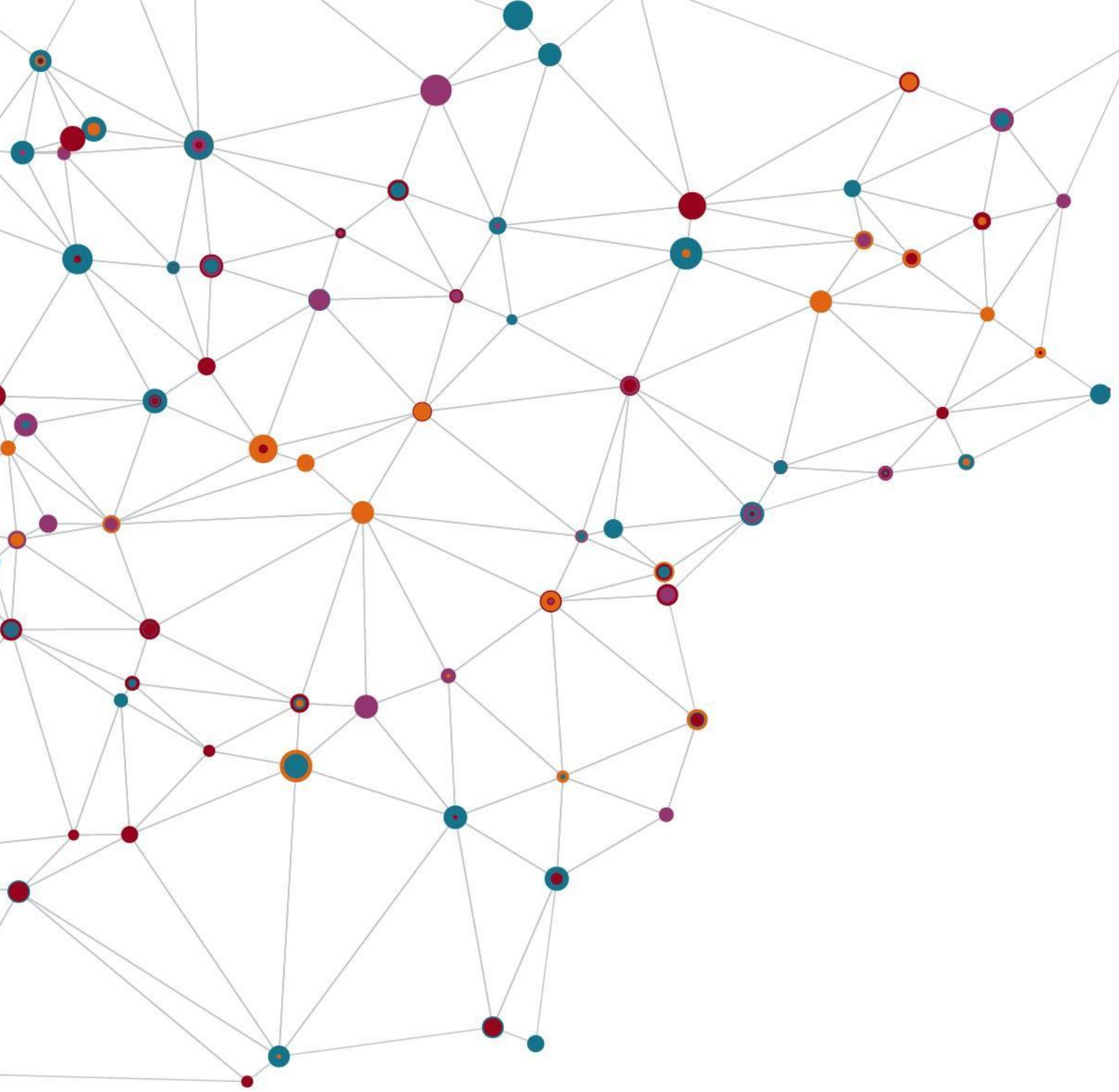


Formación Profesional

Unidad N°2

Clase 2: Python.

Objetivos.

- Reconocer las capacidades de Python.
- Entender la lógica y sintaxis básica de Python.
- Reconocer la terminología técnica del lenguaje Python.

Temas.

Introducción al Lenguaje Python. Lógica de Programación. Compiladores. Escritura de código. Operaciones aritméticas básicas. Métodos: If/else, for, while. Función.

1. Desarrollo.

Introducción a Python.

Python es un lenguaje de programación de código abierto, que hace hincapié en la legibilidad de su código. Además, es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones. Fue creado por Guido van Rossum, en Holanda, a finales de los ochentas, originalmente para reemplazar a otro lenguaje de programación conocido como ABC.

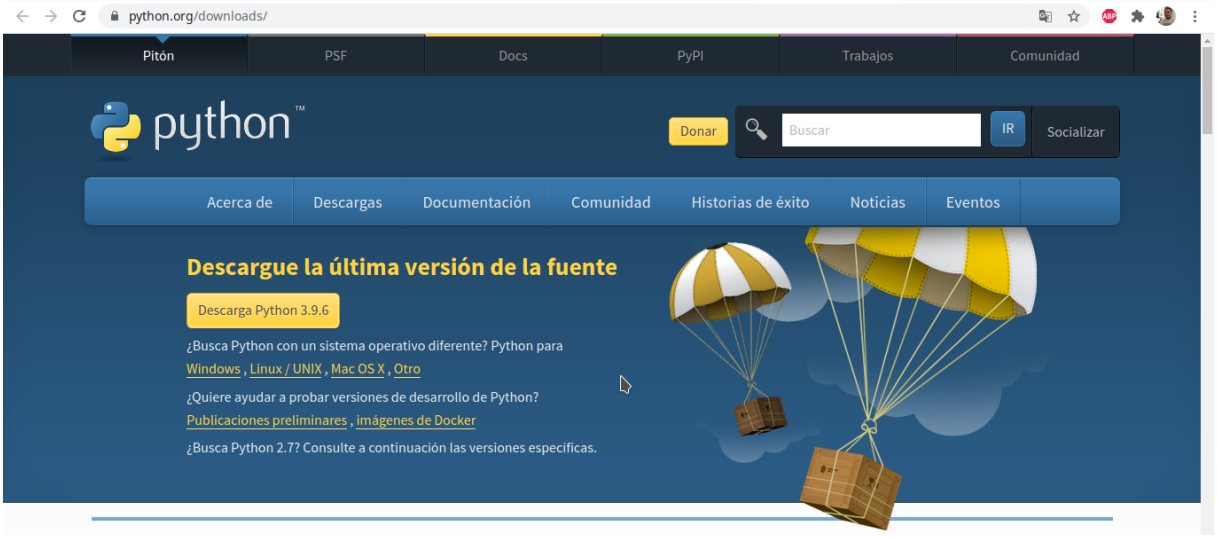
Actualmente Python se utiliza para el desarrollo de Inteligencia Artificial, para análisis de datos en el Big Data, para herramientas de desarrollo web, para desarrollo de videojuegos, y para muchísimas otras cosas, gracias a la versatilidad y adaptabilidad que le da ser de código abierto.

Programando en Python.

Antes de iniciar con la programación en Python, debemos instalar en nuestros equipos un compilador y el lenguaje a utilizar. Un compilador es un programa que se encarga de traducir el programa hecho en lenguaje de programación, a un lenguaje de máquina que pueda ser comprendido por el equipo y pueda ser procesado o ejecutado por este.

Existen muchos compiladores, y cada uno tiene sus particularidades.

Sugerimos instalar la versión más reciente de Python desde su sitio oficial: www.python.org/downloads



Una vez que Python haya sido descargado e instalado, el compilador propio de Python es llamado IDLE Python.

Otra opción válida es utilizar un compilador en línea como https://replit.com

Nuestro primer programa en Python.

Ahora que ya tenemos el Lenguaje y el compilador IDLE Python instalados, ya podemos escribir nuestro primer programa en Python. Para esto, abriremos IDLE Python y crearemos un archivo nuevo, clickeando en FILE - NEW, lo que abrirá una ventana nueva con el título UNTITLED (ya que todavía nuestro programa no tiene nombre). Esta primera ventana que se abre en el IDLE Python es el Shell de Python o la Terminal de Python, donde luego se correrán los programas que creemos.

En esta ventana es donde escribiremos nuestro código.

Al igual que con PSeint (que lo trabajamos en la unidad anterior), lo que haremos será pedirle al equipo que escriba el mensaje de HOLA MUNDO. Para esto utilizaremos la función print (). Esta función permite que toda la información que esté entre los paréntesis se imprima en pantalla.

Un detalle a tener en cuenta: Si la información que queremos que imprima en pantalla es un texto, ese texto debe estar contenido entre “”.

Nuestro primer programa debería quedar escrito de la siguiente forma:

```
print ("Hola Mundo")
```

Una vez que lo tengo escrito, debería correrlo para ver si funciona. Para correr el programa escrito debemos clickear en RUN - RUN MODULE, y antes de que el programa arranque, nos va a solicitar que lo guardemos y le asignemos un nombre (si es que no lo hicieron antes). Le asignamos el nombre hola mundo, lo guardamos, y ahí, automáticamente volveremos al Shell de Python donde se correrá el programa.

Sumando y restando.

Pasemos a programar una calculadora que sume. El código sería el siguiente:

```
print ("Esta es una calculadora que suma.")
print ("Ingrese el primer número: ")
num1 = int(input());
print ("Ingrese el segundo número: ")
num2 = int(input());
suma = num1 + num2;
print ("La suma de los números ingresados es: ",suma);
```

¿Qué está haciendo Python en esa situación?

Siempre vamos a leer de arriba hacia abajo, que es el orden en el que se ejecutan las líneas de código. La primera línea, es el mensaje que se va a imprimir, luego le solicitamos que ingrese un número por teclado y capturamos esa entrada para guardarla en la variable “num1” asegurándonos que sea un número entero. Repetimos la acción para obtener el segundo número que será guardado el la variable “num2”. Posteriormente sumamos estos números e imprimimos en pantalla el mensaje "La suma de los números ingresados es: ", para luego mostrar también el resultado de la suma que ha quedado guardada en la variable “suma". De esta forma ya tenemos nuestra calculadora para sumar 2 números.

Métodos: If/else, for, while.

Python nos permite realizar infinidad de operaciones, aritméticas, lógicas y relacionales. Muchas veces para el desarrollo de un algoritmo necesitamos repetir estas operaciones. Estos son tres métodos predefinidos que nos permitirán ser más dinámicos al programar, If/else, for, while.

En el siguiente ejemplo analizaremos el siguiente código con If/else:

```
print ("Ingrese un número: ")
num = int(input());
```

```
if num % 2 == 0:
      print("El número es par");
else:
      print("El número es impar");
```

¿Qué está haciendo Python en esa situación?

Siempre vamos a leer de arriba hacia abajo, que es el orden en el que se ejecutan las líneas de código. La primera línea, es el mensaje que se va a imprimir, con esto le solicitamos que ingrese un número por teclado. Luego capturamos esa entrada y la guardamos en la variable “num” asegurándonos que sea un número entero. Posteriormente le decimos al programa que si el resto de la división entre el número ingresado y 2 es igual a cero, que imprima en pantalla el mensaje “El número es par.”. De lo contrario, que imprima en pantalla el mensaje “El número es impar.”

Un detalle importante a tener en cuenta, es que hay que respetar la identación (identación es un espacio que corresponde a presionar la tecla TAB del teclado) en el compilador, ya que lo que le estamos pidiendo es que imprima el bloque del código que corresponde al último método ejecutado.

En el siguiente ejemplo analizaremos el siguiente código con for:

```
print("Hola Mundo");
for i in range (1,11):
      print(i)
```

¿Qué está haciendo Python en esa situación?

Recordemos que siempre vamos a leer de arriba hacia abajo, que es el orden en el que se ejecutan las líneas de código. La primera línea, es el mensaje que se va a imprimir, luego, le solicitamos que a la variable “i” le asigne secuencialmente los valores comprendidos entre los números 1 y 11, es decir, en el rango 1,11. Posteriormente le decimos que a medida que vaya leyendo y guardando dichos números, los imprima en pantalla. (atención, el último número del rango es leído, pero no impreso). Tengan en cuenta que cuando hablamos de un rango (range en Python) nos estamos refiriendo al intervalo entre el valor numérico mínimo y máximo establecido en la línea de código. Este concepto volveremos a retomarlo en próximas unidades. De esta forma obtendremos los números del 1 al 10 impresos en pantalla.

En el siguiente ejemplo analizaremos el siguiente código con while:

```
print("Hola Mundo");
fin = 0;
num = 1;
while (fin == 0):
      print(num);
      num = num + 1;
      if num > 10:
            fin = 1;
```

¿Qué está haciendo Python en esa situación?

Recordemos que siempre vamos a leer de arriba hacia abajo, que es el orden en el que se ejecutan las líneas de código. La primera línea, es el mensaje que se va a imprimir, luego, le solicitamos que a la variable “fin” le asigne el valor 0 y a la variable “num” le asigne el valor 1. Posteriormente le decimos que mientras la variable “num” sea igual a cero, realice tres acciones.

La primera es que imprima en pantalla la variable “num”.

La segunda es que a la variable “num” le sume 1.

Y la tercera, es que verifique si la variable “num” es mayor a 10. Dentro de esta última acción puede ocurrir una de dos cosas, que “num” sea menor o igual a 10, en cuyo caso el programa se vuelve a ejecutar desde el método while; o que “num” efectivamente sea mayor que 10, en ese caso la variable “fin” pasa a valer 1. Después de esto el programa se vuelve a ejecutar desde el método while pero la diferencia es que como la variable “fin” ya no es igual a cero, se da por finalizado el programa.

De esta forma obtendremos los números del 1 al 10 impresos en pantalla.

Entrada, Salida y Conversión de Datos en Python.

Python, como cualquier otro lenguaje de programación, permite a los ordenadores realizar operaciones, y que las mismas arrojen diferentes resultados.

Ya hemos impreso texto en pantalla, con la instrucción “print”, y también hemos capturado ingresos por teclado con la instrucción “input”; esto último lo volveremos a ver para dejar en claro que es todo lo que hicimos.

Para poder ingresar un número dentro de un código en Python, como ya lo hemos hecho, primero deberemos definir qué tipo de número es el que vamos a ingresar. La instrucción que utilizamos para esto es “input”, pero la línea de código entera irá variando dependiendo del tipo de número a ingresar:

- Si el número que utilizaremos será un número entero, el comando que le corresponde es int(input(“número a ingresar”));

- Si el número que utilizaremos será un número con decimales, el comando que le corresponde es float(input(“número a ingresar”));
- Si el número que utilizaremos será un número con un rango muy amplio de decimales, el comando que le corresponde es double(input(“número a ingresar”));

Repasemos el siguiente ejemplo que anteriormente hemos programado:

```
print ("Esta es una calculadora que suma.");
print ("Ingrese el primer número: ");
num1 = int(input());
print ("Ingrese el segundo número: ")
num2 = int(input());
suma = num1 + num2;
print ("La suma de los números ingresados es: ",suma);
```

Observemos que la primera línea de código contiene un “print” que indica lo que hará nuestro programa.

Luego, le indicamos al ordenador que num1 será igual al ingreso de un número entero. Podemos reemplazar num1 por cualquier otra identificación que quisiéramos ponerle, excepto palabras “reservadas”. ¿Qué significa esto? Que num1 es una variable.

Una variable es un espacio en memoria que puede guardar un dato. Una variable sería igual a una caja. Si la caja es para guardar un zapato, debería ser de un tamaño, pero si querés guardar un televisor, la caja debe ser más grande. Con las variables pasa lo mismo.

num2 es otra variable.

Y tenemos una tercera variable que se llama suma, la cual es igual a la sumatoria de num1 y num2. Así como utilizamos la sumatoria, podríamos utilizar cualquier otra operación matemática (+ suma, - resta, * multiplicación, / división).

La línea final le pide al ordenador que imprima el resultado de la variable Suma.

Eventos, comparativas y condicionales.

El siguiente ejemplo que analizaremos incluye, además de lo que vimos en el ejemplo anterior, una condición y una comparación. Para ello, tomaremos las calculadoras que ya hemos programado y crearemos una nueva calculadora básica donde se combina el ingreso de números con el condicional “si” (if).

```
print ("Esta es una calculadora que suma y resta.");
print ("Si desea sumar, presione 1.");
print ("Si desea restar, presione 2.");
ope = int(input());
if ope == 1:
      print ("Ingrese el primer número: ");
      num1 = int(input());
      print ("Ingrese el segundo número: ");
      num2 = int(input());
      suma = num1 + num2;
      print ("La suma de los números ingresados es: ",suma);
elif ope == 2:
      print ("Ingrese el primer número: ");
      num1 = int(input());
      print ("Ingrese el segundo número: ");
      num2 = int(input());
      resta = num1 - num2;
      print ("La resta de los números ingresados es: ",resta);
else:
      print("La opción es incorrecta.")
```

La primera línea de código le indica al usuario que está a punto de utilizar una calculadora.

La segunda y tercera le permiten, mediante el ingreso de un número, 1 o 2, elegir entre dos operaciones aritméticas, suma o resta.

Con la cuarta línea le estamos indicando al ordenador que debe capturar el dato que el usuario ingrese mediante el método input(), convertirlo en entero mediante el método int() y guardarlo en memoria como una variable asignándole el nombre “ope”.

En la quinta línea utilizamos el comando if para indicarle al ordenador que debe verificar si se cumple la condición que se define seguidamente, esta es, si la variable operación es igual a 1 (operacion == 1:) .

De la sexta a la octava línea, las variables identadas indican que los datos ingresados en la variable deben ser guardados y ejecutados, según corresponda.

En la novena línea utilizamos el comando if para indicarle al ordenador que debe verificar si se cumple la condición que se define seguidamente, esta es, si la variable operación es igual a 2 (operacion == 2:).

De la décima línea en adelante, las variables identadas indican que los datos ingresados en la variable deben ser guardados y ejecutados, según corresponda.

Instrucciones : Random, while, else.

Random es una biblioteca que permite la selección aleatoria de datos dentro de un rango. Para ello, debemos seleccionar entre qué números será nuestro rango.

Lo primero que debemos hacer es llamar a la biblioteca random, para lo cual utilizaremos el comando import random. A diferencia del range, que no lee el último número, el random no lee el primer número.

Analizaremos su funcionamiento en el siguiente ejemplo:

```
import random;
num = random.randint(0 , 50);
print ("Bienvenidos al juego de Adivina el número.");
print ("Estoy pensando en un número entre el 0 y el 50. ¿Cuál es?");
fin = 0;
while (fin == 0):
      minum = int(input("Ingrese el número que creas correcto: "));

      if minum < num:

            print ("El número es más alto.");
      else:

            if(minum == num):

                  print ("Ganaste!!! el número era: ", num);
                  fin = 1;
            else:

                  print ("El número es más bajo.");
```

En la primera línea llamaremos a la biblioteca con import random.

En la segunda línea, definiremos el rango en un variable llamada num, donde mediante el random.randint le pediremos al ordenador que seleccione aleatoriamente un número entero dentro del rango que definimos y lo guarde dentro de nuestra variable. En el ejemplo, nuestro rango está entre 0 y 50.

Las siguientes 2 líneas muestran en pantalla mensajes indicando qué es lo que haremos en este ejemplo, que será adivinar el número que seleccionó el ordenador.

En la quinta línea, utilizaremos una variable de control. Significa que mientras la variable no cambie, se seguirá ejecutando el programa.

En la sexta fila nos encontramos con un while, una estructura de bucle, que le permitirá al ordenador seguir ejecutando el programa hasta que se cumpla la condición que hemos definido. La condición para para la finalización del programa, es que fin !=0, osea, que la variable que definimos en la línea anterior cambie.

En las siguientes líneas encontraremos identado dentro del while la asignación que debemos cumplir, y 3 condiciones (if), que se definen de la siguiente manera:

- Se compara el número que ingresa el usuario con el número que el ordenador seleccionó aleatoriamente, y si éste es más bajo, el ordenador indicará que el número a adivinar es más alto.
- Se compara el número que ingresa el usuario con el condicional else que, analiza la condición anterior con otro if y si éste corresponde a con el número que el ordenador seleccionó aleatoriamente imprimirá el mensaje “Ganaste!!! el número era:”, luego modificará la variable: fin=1.
- Se compara el número que ingresa el usuario con el número que el ordenador seleccionó aleatoriamente, y si éste es más alto, el ordenador indicará que el número a adivinar es más bajo.

Todo esto se seguirá repitiendo en un bucle hasta que el usuario ingrese el número correcto.

Funciones.

En programación, una función es una sección de un programa que calcula un valor de manera independiente al resto del programa.

Una función tiene tres componentes importantes:

- Los parámetros, que son los valores que recibe la función como entrada;
- El código de la función, que son las operaciones que hace la función; y
- El resultado (o valor de retorno), que es el valor final que entrega la función.

En esencia, una función es un mini programa. Sus tres componentes son análogos a la entrada, el proceso y la salida de un programa.

En Python son creadas mediante la sentencia **def**:

```
def nombre(parámetros):

    código de la función;
```

Los parámetros son variables en las que quedan almacenados los valores de entrada.

La función contiene código igual al de cualquier programa. La diferencia es que, al terminar, debe entregar su resultado usando la sentencia return.

Por ejemplo, la función para calcular una suma puede ser definida de la siguiente manera:

```
def Suma():
      print("Ingrese el primer número: ");
      num1 = int(input());
      print("Ingrese el segundo número: ");
      num2 = int(input());
      print("la suma de los números es: ",num1+num2);
def Resta():
      print("Ingrese el primer número: ");
      num1 = int(input());
      print("Ingrese el segundo número: ");
      num2 = int(input());
      print("la resta de los números es: ",num1-num2);
print ("Esta es una calculadora que suma y resta.");
print ("Si desea sumar, presione 1.");
print ("Si desea restar, presione 2.");
ope = int(input());
if ope == 1:
      Suma()
elif ope == 2:
      Resta()
else:
      print("La opción es incorrecta.")
```

La primera línea define la función con el nombre Suma(). Las 4 siguientes líneas capturan los datos que ingresa el usuario y la sexta suma los datos y los muestra por pantalla.

De la 7 a la 12 hace lo mismo pero restando.

Las tres siguientes muestran texto de presentación y piden los datos.

La siguiente captura el dato.

Las últimas líneas comparan la opción ingresada y llaman a la función correspondiente para que realice la acción y muestre el resultado por pantalla.

2. Autoevaluación

Descripción.

Te invitamos a que evalúes tu comprensión y dominio del tema tratado. Al superar el desafío se te habilitará la siguiente unidad y podrás continuar.

Instrucciones.

A continuación hallarás algunas preguntas que pondrán a prueba tus saberes. Para pasar esta etapa es importante que hayas leído la unidad y practicado con los ejercicios que propuso la misma.

Ejercitación.

Ya hemos escrito nuestro primer programa. Ahora es tiempo de probar los conocimientos adquiridos.

1. En este ejercicio deberás escribir un código que te presente, por ejemplo diciendo "Hola, mi nombre es José y me gusta programar". ¿Cómo escribirías ese pequeño programa?

Rtas:

a) print ("texto de referencia");
b) print (texto de referencia);
c) print "texto de referencia";

2. ¿Cómo deberíamos escribir un código que muestre los números entre el 1 y el 10? Recordemos siempre poner una leyenda sobre lo que hace el programa. Por favor, completen con el comando faltante en la línea.

```
print("Contamos del 1 al 10");

…….("1");

…….("2");

…….("3");

…….("4");

…….("5");

…….("6");

…….("7");

…….("8");
```

```
…….("9");
…….("10");
```

Rtas:

a) Escribir
b) Print
c) print

3. ¿Cómo deberíamos escribir un código que cuente números entre el 15 y el 25? Recordemos siempre poner una leyenda sobre lo que hace el programa. Por favor, completen con el comando faltante en la línea.

```
print("Contamos del 15 al 25")
for i in ……. (15,26):
      print(i)
```

Rtas:

a. rango
b. number
c. range

4. ¿Cómo deberíamos escribir un código que cuente números entre el 15 y el 25? Recordemos siempre poner una leyenda sobre lo que hace el programa. Por favor, completen con el comando faltante en la línea.

```
print("Contamos del 15 al 25");
fin = 0;
num = 1;
……. (fin == 0):
      print(num);
      num = num + 1;
      if num > 10:
            fin = 1;
```

Rtas:

a. Wile
b. While
c. while

5. ¿Cómo deberíamos escribir un código que nos diga si un número es mayor que 2 o no? Recordemos siempre poner una leyenda sobre lo que hace el programa. Por favor, completen con el comando faltante en la línea.

```
print ("Ingrese un número para saber si es mayor a dos: ")
num = int(input());
... num>2:
      print("El número es mayor que 2.");
else:
      print("El número es igual o menor que 2.");
```

Rtas:

a. If
b. else
c. if

6. En este ejercicio deberás escribir un código que te permita restar 2 números con dos decimales. ¿Cómo escribirías ese pequeño programa? No olvides poner un mensaje que indique lo que estamos haciendo. Por favor, completa las líneas de código con la información faltante:

```
print("Este es un programa para restar dos números con 2 decimales")
num1 = ……. (input("Ingrese el primer número: "))
num2 = ……. (input("Ingrese el segundo número: "))
Resta = Num1 - Num2
print("La resta de los números es: " , Resta)
```

Rtas:

a. int
b. float
c. dec

7. Ya armamos una calculadora que realiza operaciones de Suma y de Resta. Ahora debemos completarla con Multiplicación y División. Por favor, completar la información faltante en las líneas. Recuerden que los símbolos para operaciones matemáticas son: + suma, - resta, * multiplicación y / división.

```
print("Esta es una calculadora.")
print("Ingrese 1 para suma.")
```

```
print("Ingrese 2 para resta.")
print("Ingrese 3 para multiplicar.")
print("Ingrese 4 para dividir.")
operacion = int(input())
if operacion == 1:
      num1 = float(input("Ingrese el primer número a sumar: "))
      num2 = float(input("Ingrese el segundo número a sumar: "))
      suma = num1 ….. num2
      print("la suma es: " , suma)
if operacion == 2:
      num1 = float(input("Ingrese el minuendo: "))
      num2 = float(.......("Ingrese el sustraendo: "))
      resta = num1 - num2
      print("la resta es: " , resta)
if operacion == 3:
      num1 = float(input("Ingrese el primer número a multiplicar: "))
      num2 = float(input("Ingrese el multiplicador: "))
      multi = num1 ……. num2
      print("la multiplicación es: " , multi)
if operacion == 4:
      num1 = float(input("Ingrese el primer número a dividir: "))
      num2 = …… (input("Ingrese el divisor: "))
      divi = num1 / num2
      print("el resultado es: " , divi)
```

Rtas:

a. - ; input ; / ; int
b. + ; input ; / ; float
c. + ; input ; * ; float

3. Tarea.

1. Programa una calculadora en Python para cada una de las cuatro operaciones aritméticas básicas.

2. Diseña un juego de lotería empezando por el diagrama de flujo, pasando por el pseudocódigo y terminando con un programa en Python.

3. Diseña una calculadora como la vista en el ejercicio 2 y mejórala lo más que puedas agregando funciones.

4. A la calculadora que acabas de programar, añádele la posibilidad de que el usuario juegue a la lotería.

5. En ningún caso el programa se debe cerrar a menos que el usuario así lo decida.