**UNIVERSIDAD DE GUADALAJARA**

**CUCEI**

**DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN**

**DEPARTAMENTO DE CIENCIAS COMPUTACIONALES**

**TAREA NO. 3**

*TEMA***: EXPRESIÓN ARITMÉTICA, OPERADORES ARITMÉTICOS, SU PRIORIDAD, TIPOS DE DATOS Y VARIABLE**

*NOMBRE DEL ESTUDIANTE*: **ROBLES PULIDO EFRAIN**

*NOMBRE DE LA MATERIA:* **FUNDAMENTOS FILOSOFICOS DE LA COMPUTACION**

*SECCIÓN:* **D13**  *CALENDARIO:* **2021A**

*NOMBRE DEL PROFESOR:* **LUIS FELIPE MARISCAL LUGO**





**Marco Teórico**

* **Expresión aritmética**

Las expresiones aritméticas combinan los operadores con datos numéricos y generan un nuevo número como resultado. Por ejemplo, si se tiene la base y la altura de un triángulo, mediante una expresión aritmética se puede obtener su área.

Los operadores aritméticos se utilizan para operar tanto datos de tipo enteros como reales.

La combinación de diferentes operadores aritméticos en una misma expresión puede hacer que ésta resulte ambigua; es decir, que sea posible más de una interpretación y por ende más de un resultado.

Ejemplo:

Declaración de variables

Real: base, altura, área

Asignación de valores

base = 10

altura = 15

Expresión aritmética para calcular el área:

base \* altura / 2;

Asignación del resultado de una expresión aritmética a una variable:

área = base \* altura / 2

En resultado de esta expresión es 75 y se guarda a la variable área.

Una expresión matemática realiza un cálculo y da un valor. El siguiente es un ejemplo de una expresión matemática simple:

12 + 2

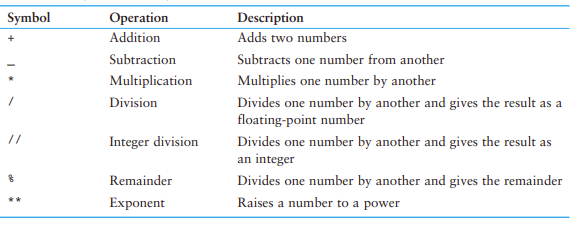
Los valores a la derecha e izquierda del operador + se denominan operandos. Estos son valores que el operador + suma. Si escribe esta expresión en modo interactivo, verá que da el valor 14:

12 + 2 e

14

Las variables también se pueden usar en una expresión matemática. Cuando usamos una expresión matemática para calcular un valor, normalmente queremos guardar ese valor en la memoria para que podamos usarlo nuevamente en el programa. Hacemos esto con una declaración de asignación.

* **Tabla de los operadores aritméticos**



* **Prioridad de los operadores**

Los operadores tienen un orden jerárquico que determina cuál se ejecuta primero cuando hay varios en la misma expresión. Para alterar el orden de ejecución determinado por la jerarquía es necesario utilizar paréntesis.

Primero, las operaciones que están entre paréntesis se realizan primero. Luego, cuando dos operadores comparten un operando, se aplica primero el operador con mayor precedencia. La precedencia de los operadores matemáticos, de mayor a menor, es:

1. Exponenciación: \*\*

2. Multiplicación, división y resto: \* /, //, %

3. Suma y resta:

Observe que la multiplicación (\*), la división de punto flotante (/), la división de enteros (//) y los operadores de resto (%) tienen la misma precedencia. La suma () y la resta () los operadores también tienen la misma precedencia. Cuando dos operandos con la misma precedencia comparten un operando, los operadores se ejecutan de izquierda a derecha.

Ejemplos:

3 \* 2 + 5 = 6 + 5 = 11

3 \* (2 +5) = 3 \* 7 = 21

6 + 4 / 2 = 6 + 2 = 8

(6 + 4) / 2 = 10 / 2 = 5

5 \* 3 + 8 / 2 – 1 = 15 + 4 - 1 = 18

5 \* (3 + 8) / (2 - 1) = 5 \* 11 / 1 = 55

Las partes de una expresión matemática pueden agruparse con paréntesis para forzar que algunas operaciones se realicen antes que otras. En la siguiente declaración, las variables a y b son sumados, y su suma se divide por 4:

resultado = (a + b) / 4

Sin embargo, sin los paréntesis, b se dividiría entre 4 y el resultado se agregaría a a.

* **Tipo de datos**

Debido a que los diferentes tipos de números se almacenan y manipulan de diferentes maneras, Python utiliza tipos de datos para categorizar valores en la memoria.

Python tiene cinco tipos de datos estándar:

* + Números

Los tipos de datos numéricos almacenan valores numéricos. Los objetos numéricos se crean cuando se les asigna un valor. Python soporta cuatro tipos numéricos diferentes:

* int (enteros con signo)
* long (enteros largos, también se pueden representar en octal y hexadecimal)
* float (valores reales de punto flotante)
* complex (números complejos)
  + Cadena (Strings)

Las cadenas en Python se identifican como un conjunto contiguo de caracteres representados en las comillas. Python permite pares de comillas simples o dobles. Los subconjuntos de cadenas se pueden tomar usando el operador de segmento ([] y [:]) con índices que comienzan en 0 al principio de la cadena y avanzan desde -1 al final.

El signo más (+) es el operador de concatenación de cadenas y el asterisco (\*) es el operador de repetición.

Por ejemplo:

#!/usr/bin/python

str = 'Hola mundo!'

print str # Imprime una cadena completa

print str [0] # Imprime el primer carácter de la cadena

print str [2: 5] # Imprime caracteres desde el 3 al 5

print str [2:] # Imprime la cadena a partir del 3er carácter

print str \* 2 # Imprime cadena dos veces

print str + "TEST" # Imprime una cadena concatenada

*Esto producirá el siguiente resultado:*

Hola Mundo!

H

la

la mundo!

Hola mundo! Hola mundo!

Hola mundo! TEST

* + Lista

Una lista contiene elementos separados por comas y encerrados entre corchetes ([]). Hasta cierto punto, las listas son similares a las matrices en C. Una diferencia entre ellas es que todos los elementos que pertenecen a una lista pueden ser de diferentes tipos de datos.

Se puede acceder a los valores almacenados en una lista utilizando el operador de división ([] y [:]) con índices que comienzan en 0 al principio de la lista y continúan hasta el final -1. El signo más (+) es el operador de concatenación de lista, y el asterisco (\*) es el operador de repetición.

* + Tupla

Consiste en una serie de valores separados por comas. A diferencia de las listas, sin embargo, las tuplas están encerradas entre paréntesis.

Las principales diferencias entre las listas y las tuplas son: Las listas están encerradas entre corchetes ([]) y sus elementos y tamaño se pueden cambiar, mientras que las tuplas se encierran entre paréntesis (()) y no se pueden actualizar. Las tuplas se pueden considerar como listas de solo lectura.

* + Diccionario

Son un tipo de tabla hash. Funcionan como matrices asociativas o hashes y consisten en pares clave-valor. Una clave de diccionario puede ser de casi cualquier tipo de dato en Python, pero generalmente son números o cadenas. Los valores, por otro lado, pueden ser cualquier objeto Python arbitrario.

Los diccionarios están encerrados entre llaves ({}) y los valores pueden asignarse y accederse utilizando llaves ([]).

* **Tipos de variables**

Las variables no son más que ubicaciones de memoria reservadas para almacenar valores. Esto significa que, al crear una variable, se reserva algo de espacio en la memoria.

Según el tipo de datos de una variable, el intérprete asigna memoria y decide qué se puede almacenar en la memoria reservada. Por lo tanto, al asignar diferentes tipos de datos a las variables, puede almacenar números enteros, decimales o caracteres en estas variables.

El objeto puede ser de alguno de los tipos vistos (número o cadena de texto), o alguno de los otros tipos existentes en Python.

Cada variable debe tener un nombre único llamado identificador. Eso es muy de ayuda pensar las variables como contenedores que contienen data el cual puede ser cambiado después a través de técnicas de programación.

**Bibliografía:**

Comenzando con Python (3ª edición) PDF - Firebase Python Programming (2ª edición) Inventa sus propios juegos de computadora con Python: ... Hacking: Hacking Made Easy 1: Principiantes: - [Documento PDF]. (2021). Consultado el 13 de marzo de 2021 en <https://fdocuments.in/document/starting-out-with-python-3rd-edition-pdf-firebase-python-programming-2nd-edition.html>

Elementos de programación. (2021). Consultado el 13 de marzo de 2021 en <https://core.ac.uk/download/pdf/322589792.pdf>

3.2. Variables y constantes - Materiales del entrenamiento de programación en Python - Nivel básico. (2021). Consultado el 13 de marzo de 2021 en <https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion3/variables_constantes.html>

Tipos de Variables en Python - Código Fuente. (2021). Consultado el 13 de marzo de 2021 en <https://www.codigofuente.org/variables-en-python/>

**Resolver los problemas**

**1.-Definición del problema**

Solicitar al usuario dos números enteros, aplicar todos los operadores aritméticos (+,-,\*,/, //,%,\*\*), guardar el resultado en un tercer variable (se utilizará la misma variable para guardar cada uno de los resultados), por último imprimir el resultado de cada operación. Observa la siguiente imagen para ayudarte a entender mejor el problema a resolver.

**Desarrollar el análisis**

**Entrada:** Solicitar al usuario dos datos de tipo entero, vamos a utilizar dos variables, en cada una de ellas almacenar un dato.

**Proceso:** Vamos a utilizar los operadores aritméticos (suma, resta, multiplicación, división, división entera, residuo de división y potencia) y el operador de asignación ( ).

**Salida:** Mostrar el resultado en pantalla de cada operación aritméticos con los dos datos de entrada, es decir, imprimir el contenido de las variables de cada operación aritmética.

**Diseño del algoritmo**

**Diagrama de flujo**

**Diagrama

Descripción generada automáticamente**

**Pseudocódigo**

#Efrain Robles Pulido

#Algoritmo para utilizar las operaciones aritméticas con dos valores y mostrar el resultado de cada uno

#Entrada

Entrada (“Dame un número:”)

Leer (n1)

Entrada (“Dame otro número:”)

Leer (n2)

#Proceso

s n1+n2

r n1-n2

m n1\*n2

d n1/n2

de n1//n2

rd n1%n2

p n1\*\*n2

#Salida

Imprimir (“El resultado de la suma es = “, s)

Imprimir (“El resultado de la resta es = “, r)

Imprimir (“El resultado de la multiplicación es = “, m)

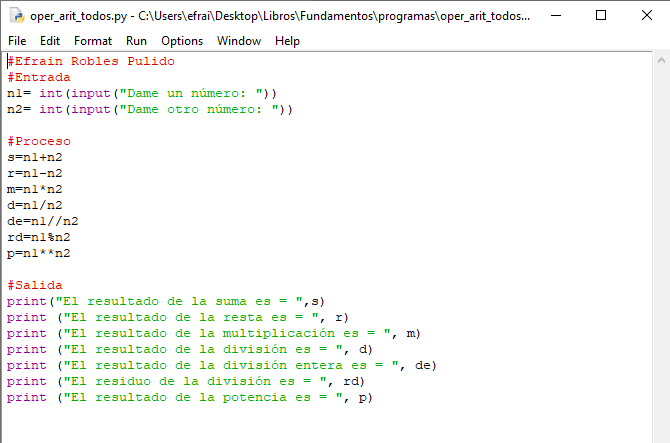
Imprimir (“El resultado de la división es = “, d)

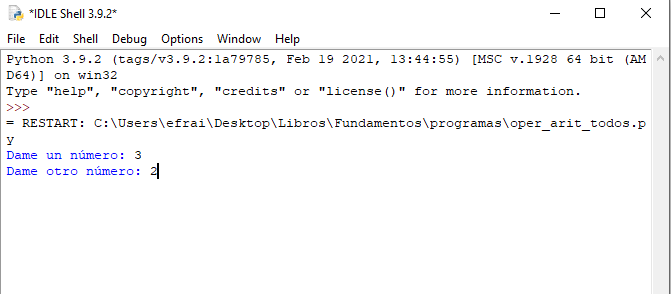
Imprimir (“El resultado de la división entera es = “, de)

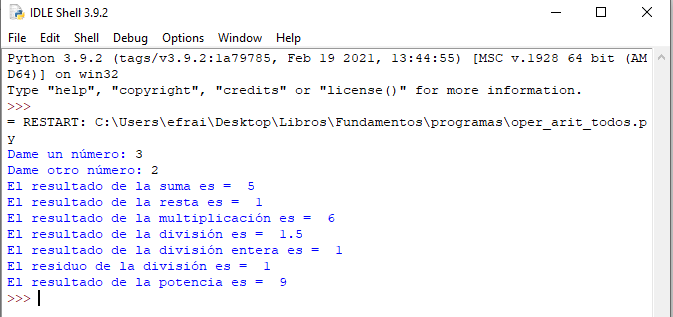
Imprimir (“El residuo de la división es = “, rd)

Imprimir (“El resultado de la potencia es = “, p)

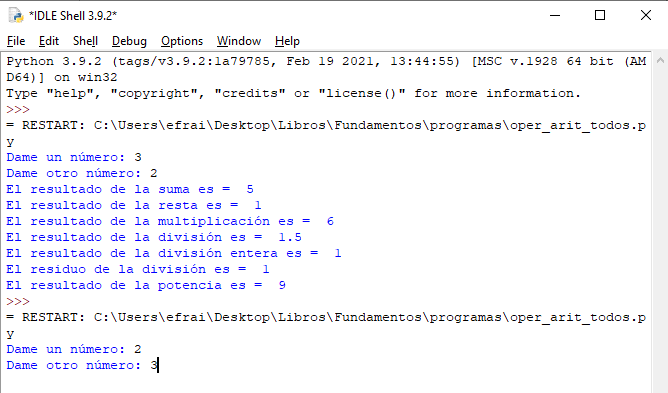
**Codificación**

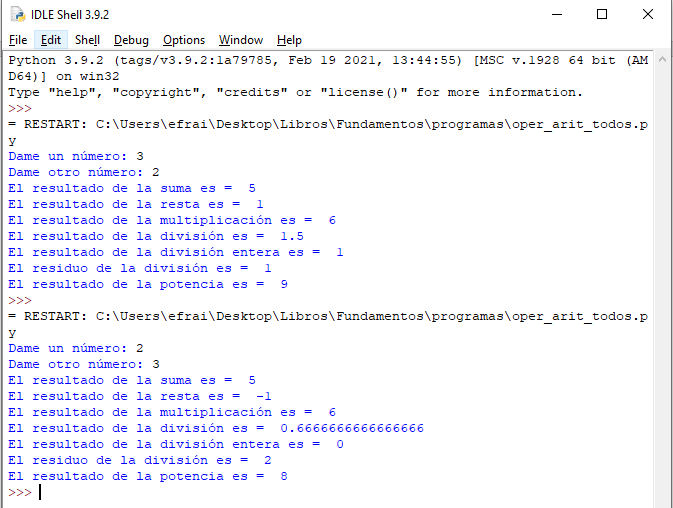






**Caso #2**





**2.-Definición del problema**

Solicitar al usuario tres números enteros, calcular el promedio, guardar el resultado en una cuarta variable, por último, imprimir el resultado. Observa la siguiente imagen para ayudarte a entender mejor el problema a resolver.

**Desarrollar el análisis**

**Entrada:** Solicitar al usuario tres datos de tipo entero, vamos a utilizar tres variables, en cada una de ellas almacenar un dato.

**Proceso:** Vamos a utilizar los operadores aritméticos (suma y división) y el operador de asignación ( ).

**Salida:** Mostrar el resultado en pantalla, es decir, imprimir el contenido de la variable p.

**Diseño del algoritmo**

**Diagrama de flujo**

**Diagrama

Descripción generada automáticamente**

**Pseudocódigo**

#Efrain Robles Pulido

#Algoritmo para sumar tres números enteros y mostrar el resultado

#Entrada

Entrada (“Dame la primera calificación:”)

Leer (n1)

Entrada (“Dame la segunda calificación: ”)

Leer (n2)

Entrada (“Dame la tercera calificación: ”)

Leer (n3)

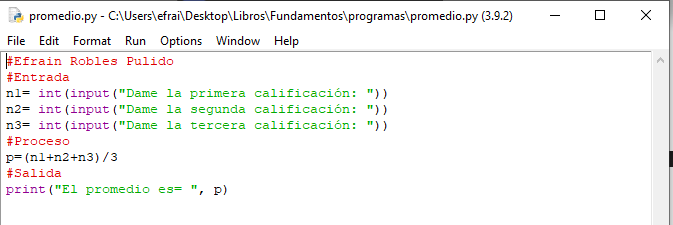
#Proceso

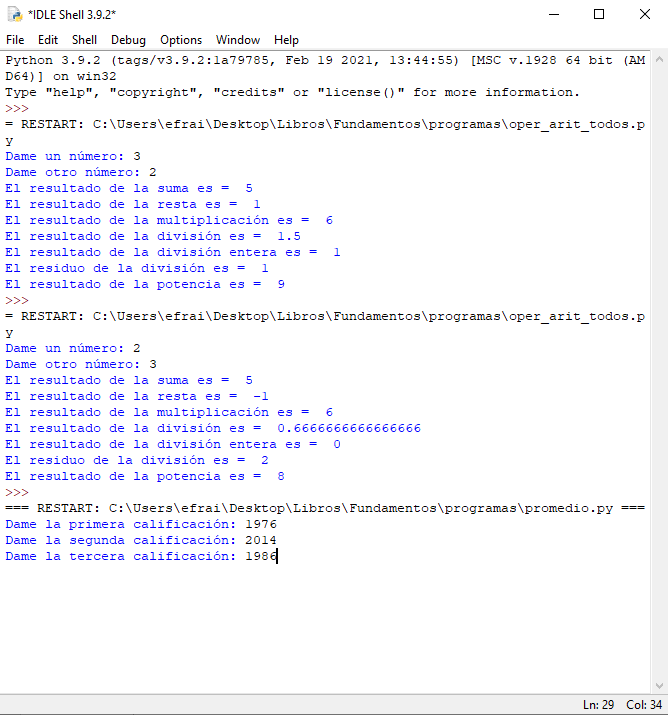
p (n1+n2+n2)/3

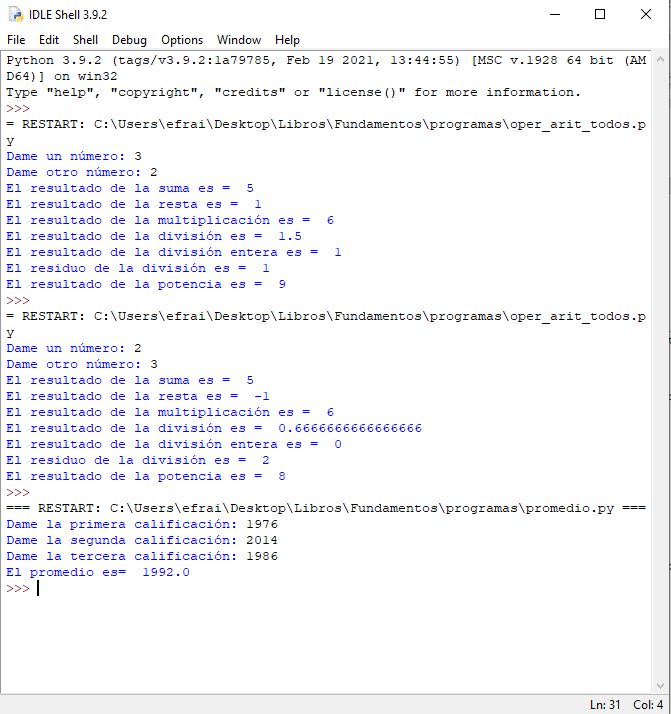
#Salida

Imprimir (“El promedio es = “, p)

**Codificación**







**Conclusión:**

En esta tarea aprendí lo básico para poder manejar las operaciones aritméticas en Python, mediante la aplicación de este conocimiento en el programador de Python aprendí como es las estructuras, sintaxis y el seguimiento de las operaciones para que sean correctas los cálculos en el código. También aprendí como se debe de empezar a programar, desde el análisis del problema, el diagrama de flujo y el seudocódigo para luego pasar la información recopilada a un código donde tendrá la sintaxis correcta de Python.