

Laboratorio 1 Parcial 2: Ejercicios en Python

Ednan Josué Merino Calderón^[L00405925]

Universidad de las Fuerzas Armadas - ESPE
ejmerino@espe.edu.ec

Abstract.

Mediante el lenguaje de programación Python se realizó veinte aplicativos simples dentro de un mismo código, mediante la ayuda de un menú selector que ayudará al usuario a elegir la opción que requiera. El mismo programa se encuentra debidamente comentado y desarrollado mediante los convenios de programación, definiendo los nombres de las clases con "snake_case"

1 Introducción

Python es un lenguaje de programación interpretado tipificado dinámicamente cuya filosofía enfatiza una sintaxis que facilita el código legible. Es un lenguaje de programación multiparadigma disponible en varias plataformas. En otras palabras, Python se ejecuta sin el procesamiento del compilador y los errores se detectan en tiempo de ejecución. Admite programación funcional, programación imperativa y programación orientada a objetos. Las variables se comprueban en tiempo de ejecución. Está disponible para plataformas Windows, Linux o MAC y no tiene licencia de programación. Python se usa porque lo usan grandes empresas, es fácil de aprender y mantener, es gratuito y de código abierto, incluye una gran cantidad de bibliotecas y tiene una gran comunidad. Al utilizar una sintaxis legible, la curva de aprendizaje es muy rápida.

2 Método

Las opciones para escoger dentro del menú son por lo general problemas matemáticos para estudiantes de Educación General Básica superior, Bachillerato General Unificado y en algunos casos Primeros Niveles de Educación Superior. Por lo que este aplicativo puede ir destinado, por lo general - más allá de la instructora encargada de calificar el software - a los estudiantes y jóvenes entre 15 y 24 años de edad. Al ser problemas matemáticos y resolución de fórmulas, se entiende que las respuestas, están debidamente validadas y mantienen lógica y coherencia con lo pedido.

El siguiente trabajo, comentado y documentado se encuentra en el conocido sistema de versionamiento GitHub: https://github.com/ejmerino/ModelosDiscretos/blob/main/%5BNRC_8001%5D_Lab1Unidad2_MerinoCalderon_EdnanJosue.py

A continuación se especificará los veinte ejercicios en Python desarrollados por el autor. Para evitar la desprolijidad y el desorden, se implementaron los veinte programas dentro de un solo aplicativo, dominando por un menú principal. Cada dato ingresado por el usuario está debidamente validado por una sentencia try/except en cada uno de las funciones. Los programas escogidos y desarrollados fueron los siguientes:

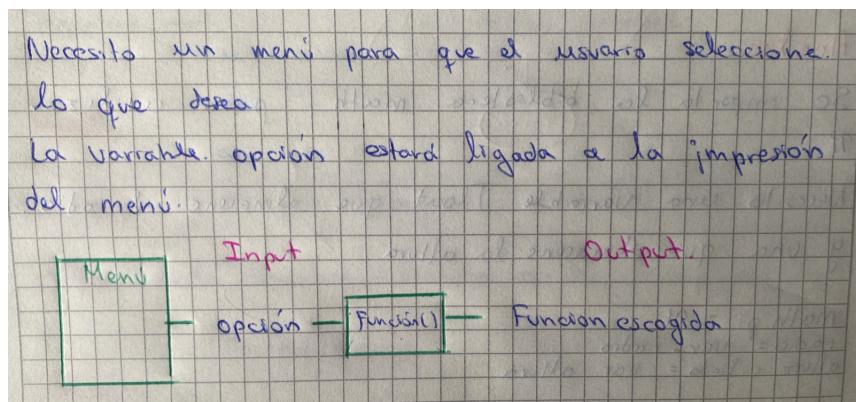
1. Ingrese su nombre y sea saludado
2. Calcular el Perímetro y Área del Paralelogramo
3. Calcular volúmen de un cilindro
4. Calcular el área de un círculo
5. Cálculo de potencias x a la y
6. Calcular Volúmen de una esfera
7. Calcular el área de un rectángulo
8. Teorema de Pitágoras
9. Calcular la expresión $y = x$ a la z entre 2
10. Calcular la rapidez
11. Calculo densidad de un objeto
12. Convertir días a años/meses/semanas
13. Transformar cm a m y km
14. Transformar de °F a °C o de °C a °F
15. Convertir libras a kilos y gramos
16. Hallar la raíz cuadrada de un número
17. Hallar el ángulo faltante de un triángulo
18. Hallar área de un triángulo
19. Operaciones básicas con dos números
20. Hallar datos de un círculo

3 Resultados y Análisis

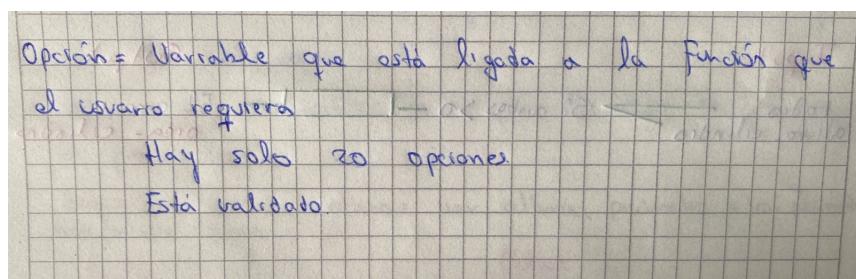
3.1 Menú Principal

Para empezar se definió el menú anteriormente especificado, donde se encontraban todas las opciones a ser elegidas por el usuario, y cuidadosamente enumeradas anteriormente.

Análisis El análisis es el siguiente:

**Fig. 1.** Análisis Menú

Modelo El modelo a continuación:

**Fig. 2.** Caption

Programación Obedeciendo al análisis y modelo después de ser desplegado el Menú, el usuario debo elegir una función.

```

1 def menu():
2     opcion = ""
3     while opcion != "21":
4         menu = """
5             LABORATORIO 1 PARCIAL 2 POR JOSUÉ MERINO
6
7             1. Ingrese su nombre
8             2. Calcular el Perímetro y Área del Paralelogramo
9             3. Calcular volumen de un cilindro
10            4. Calcular el área de un círculo
11            5. Cálculo de potencias x^y
12            6. Calcular Volumen de una esfera
  
```

```
13    7. Calcular el área de un rectángulo
14    8. Teorema de Pitágoras
15    9. Calcular la expresión  $y=(x^z)/2$ 
16   10. Calcular la rapidez
17   11. Calculo densidad de un objeto
18   12. Convertir días a años/meses/semanas
19   13. Transformar cm a m y km
20   14. Transformar de F a C o de C a F
21   15. Convertir libras a kilos y gramos
22   16. Hallar la raíz cuadrada de un número
23   17. Hallar el ángulo faltante de un triángulo
24   18. Hallar área de un triángulo
25   19. Operaciones básicas con dos números
26   20. Hallar datos del círculo
27   21. Salir
28
29 Elija su opción: """
30 opcion = input(menu)
31 if opcion=="1":
32     saludo()
33     break
34 if opcion == "2":
35     paralelogramo()
36     break
37 if opcion == "3":
38     cilindro()
39     break
40 if opcion == "4":
41     area_circulo()
42     break
43 if opcion == "5":
44     potencia()
45     break
46 if opcion=="6":
47     volumen_esfera()
48     break
49 if opcion=="7":
50     area_perimetro_rectangulo()
51     break
52 if opcion=="8":
53     pitagoras_menu()
54     break
55 if opcion=="9":
56     calculo_expresion()
57     break
58 if opcion=="10":
59     velocidad_fisica()
60     break
61 if opcion=="11":
62     calculo_densidad()
```

```
63         break
64     if opcion=="12":
65         conversion_dias()
66         break
67     if opcion=="13":
68         conversion_longitud()
69         break
70     if opcion=="14":
71         menu_temperatura()
72         break
73     if opcion=="15":
74         convertir_libras()
75         break
76     if opcion=="16":
77         raiz_cuadrada()
78         break
79     if opcion=="17":
80         encontrar_angulo()
81         break
82     if opcion=="18":
83         area_triangulo()
84         break
85     if opcion=="19":
86         aritmetica()
87         break
88     if opcion=="20":
89         circulo()
90         break
```

Programacion

Casos de Prueba y Ejecución del programa La ejecución del programa es la siguiente:

```

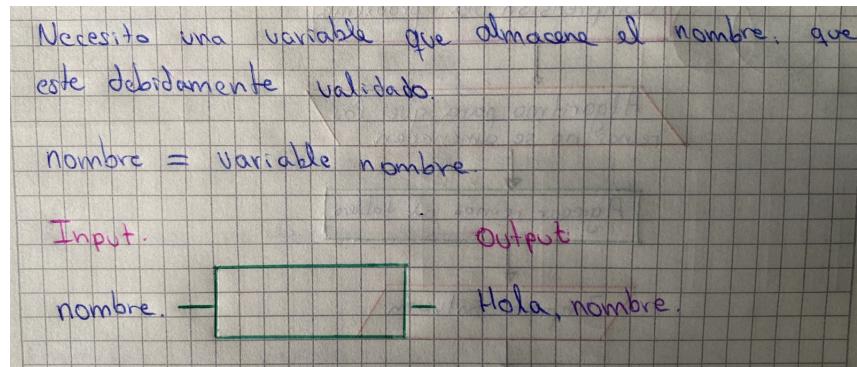
LABORATORIO 1 PARCIAL 2 POR JOSUÉ MERINO
-----
1. Ingrese su nombre
2. Calcular el Perímetro y Area del Paralelogramo
3. Calcular volumen de un cilindro
4. Calcular el área de un círculo
5. Cálculo de potencias x^y
6. Calcular Volúmen de una esfera
7. Calcular el área de un rectángulo
8. Teorema de Pitágoras
9. Calcular la expresión y=(x^z)/2
10. Calcular la rapidez
11. Calculo densidad de un objeto
12. Convertir días a años/meses/semanas
13. Transformar cm a m y km
14. Transformar de °F a °C o de °C a °F
15. Convertir libras a kilos y gramos
16. Hallar la raiz cuadrada de un número
17. Hallar el ángulo faltante de un triángulo
18. Hallar área de un triángulo
19. Operaciones básicas con dos números
20. Hallar datos del círculo
21. Salir
-----
Elija su opción:

```

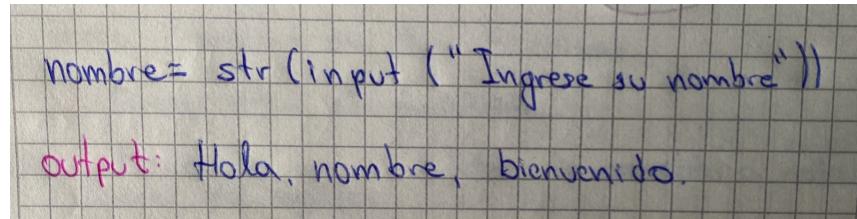
Fig. 3. Ejecución Menú Principal

3.2 Ingrese su nombre y sea saludado

Análisis El análisis del problema es el siguiente:

**Fig. 4.** Análisis Saludo

Modelo El modelo del programa a continuación:

**Fig. 5.** Modelo Saludo

Programación En esta función existe solamente un parámetro, la variable nombre, que es un dato ingresado por el usuario por teclado. Se retorna la solución que viene siendo el saludo por parte del programa hacia el usuario.

```

1 def saludo():
2     """
3         Función que sirve para pedirle el nombre al usuario y
4         saludarlo
5     """
6     Parámetros:
7     nombre: dato str ingresado por el usuario
8
9     Retorna:
10    Saludo
11    """
12    #Bucle para la variable nombre
13    while True:
14        #Sentencia try/except para validar datos
15        try:
16            #El usuario ingresa su nombre y se almacena en la
17            #variable nombre
18            nombre=str(input("Ingrese su nombre:"))
19            #break
20            break
21        #Si el usuario no ingresó un tipo de dato str
22        except ValueError:
23            #Se le comunica al usuario que no ingresó un dato
24            #válido
24            print("Ingrese un dato válido!")
25        #Se imprime la solución
26        print("Hola",nombre,"Bienvenido :)")

```

Programacion

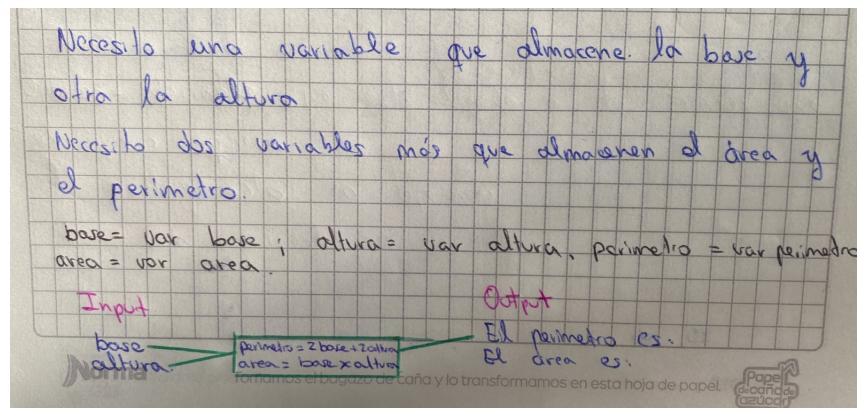
Casos de Prueba y Ejecución del programa La ejecución del programa es la siguiente:

```
Elija su opción: 1
Ingrese su nombre:Ednan Josué
Hola Ednan Josué Bienvenido :)
```

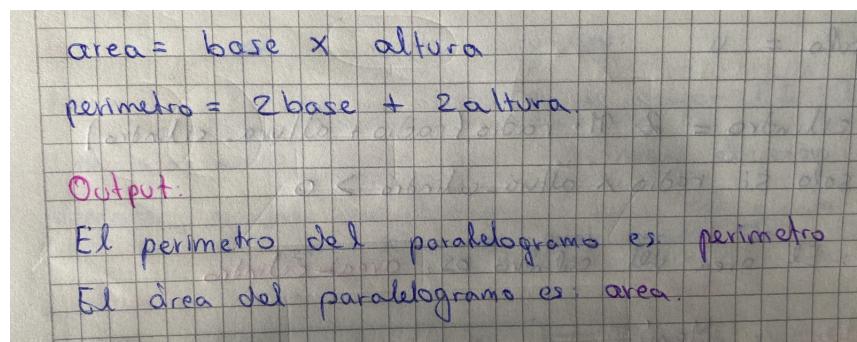
Fig. 6. Ejecución Saludo

3.3 Calcular el Perímetro y Área del Paralelogramo

Análisis El Análisis es el siguiente:

**Fig. 7.** Análisis Paralelogramo

Modelo El Modelo a continuación:

**Fig. 8.** Modelo Paralelogramo

Programación Siguiendo el Modelo y el análisis se le pide al usuario que ingrese la base y la altura de un paralelogramo.

```
1 def paralelogramo():
2     """
3         Función Que Calcula el área y perímetro de un
4         paralelogramo
5         _____
6         Parámetros:
7             Base= Número float ingresado por el usuario
8             Altura= Número float ingresado por el usuario
9         _____
10        Retorna:
11            Area y Perímetro de un paralelogramo
12        _____
13        """
14    #Se inicia un bucle
15    while True:
16        #Sentencia try/except para validar datos
17        try:
18            #El usuario ingresa un tipo de dato float y se
19            #almacena en la variable base
20            base = float(input("Ingrese la base:"))
21            #break
22            break
23        #Si el usuario no ingresó un tipo de dato float
24        except ValueError:
25            #Se le notifica al usuario que ingresó un dato
26            #incorrecto
27            print("Ingrese un dato válido!")
28    #Se inicia un bucle
29    while True:
30        #Sentencia try/except para validar datos
31        try:
32            #El usuario ingresa un tipo de dato float y se
33            #almacena en la variable altura
34            altura = float(input("Ingrese la altura:"))
35            #break
36            break
37        #Si el usuario no ingresó un tipo de dato float
38        except ValueError:
39            #Se le notifica al usuario que ingresó un dato
40            #incorrecto
41            print("Ingrese un dato válido!")
42    #Si es que el cilindro existe la base y la altura son
43    #mayores que 0
44    if base>0 and altura >0:
45        #Cálculo del perímetro
46        perimetro=altura+altura+base+base
```

```

41      #Se imprime la solución del perímetro
42      print("El perímetro del paralelogramo es:",perimetro , "
43      u")
44      #Cálculo del área
45      area=base*altura
46      #Se imprime la solución del área
47      print("El área del paralelogramo es:",area , "u^2")
48      #Si el cilindro no existe
49      else:
50          #Se le notifica al usuario
51          print("El paralelogramo no existe!")
52          #Se le notifica al usuario que lo intente de nuevo
53          print("Inténtelo de nuevo!")
54          #Se empieza de nuevo
55          paralelogramo()

```

Programacion

Casos de Prueba y Ejecución del programa Cuando el paralelogramo existe:

```

Elija su opción: 2
Ingrese la base:2
Ingrese la altura:2
El perímetro del paralelogramo es: 8.0 u
El área del paralelogramo es: 4.0 u^2

```

Fig. 9. Ejecución Paralelogramo

Cuando el paralelogramo no existe:

```

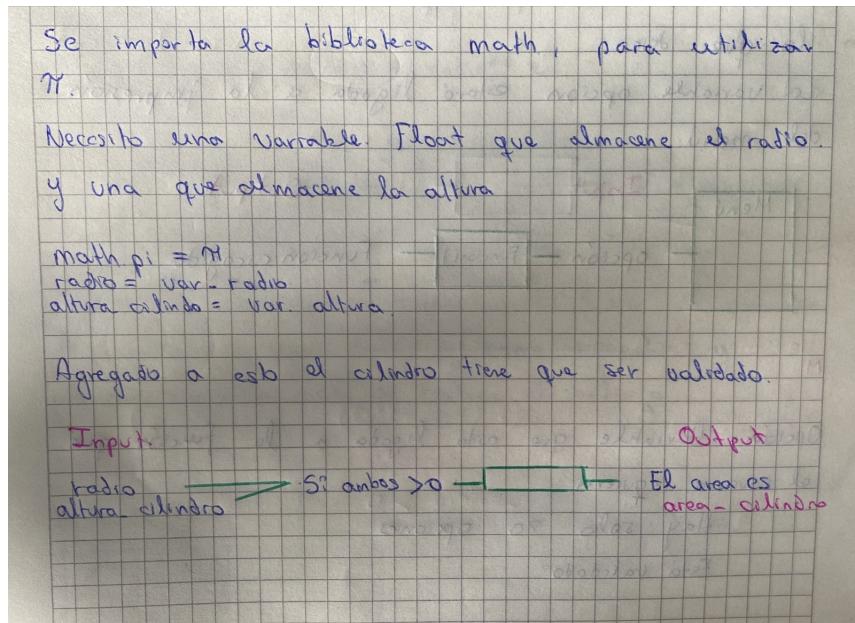
Elija su opción: 2
Ingrese la base:-5
Ingrese la altura:-1
El paralelogramo no existe!
Inténtelo de nuevo!

```

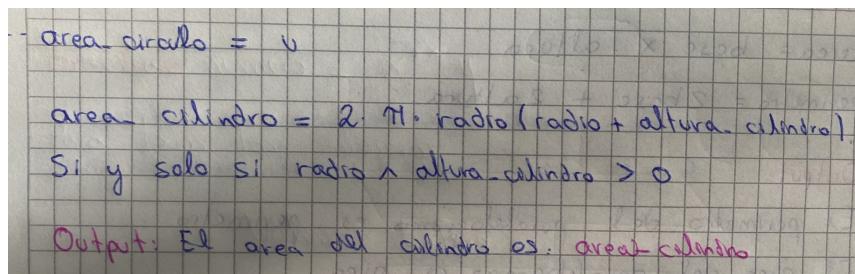
Fig. 10. Paralelogramo inválido

3.4 Volumen de Un Cilindro

Análisis EL Analysis es el siguiente:

**Fig. 11.** Análisis Cilindro

Modelo El Modelo a continuación:

**Fig. 12.** Modelo Cilindro

Programación Siguiendo el modelo y el análisis:

```

1 def cilindro():
2     """
3         Función para calcular el área de un cilindro
4
5     Parámetros:

```

```
6     radio: valor float ingresado por el usuario
7     altura_cilindro: valor float ingresado por el usuario
8
9     Retorna:
10    area_cilindro: valor en unidades cuadradas, resultado del
11        área del cilindro
12        """
13    #Se inicia un bucle
14    while True:
15        #Se emplea la sentencia try/except para validar datos
16        try:
17            #El usuario ingresa el radio de tipo float en la
18            variable radio
19            radio = float(input("Ingrese el radio:"))
20            #break
21            break
22            #Si es que el usuario ingresa cualquier otro dato que
23            no sea de tipo float
24            except ValueError:
25                #Se le notifica al usuario que ingreso un dato
26                erroneo
27                print("Ingrese un dato válido!")
28    #Se inicia otro bucle para la variable altura_cilindro
29    while True:
30        #Se emplea la sentencia try/except para validar datos
31        try:
32            #El usuario ingresa la altura de tipo float en la
33            variable altura_cilindro
34            altura_cilindro = float(input("Ingrese la altura:"))
35            #break
36            break
37            #Si es que el usuario ingresa cualquier otro dato que
38            no sea de tipo float
39            except ValueError:
40                #Se le notifica al usuario que ingreso un dato
41                erroneo
42                print("Ingrese un dato válido!")
43    #Si es que el radio y altura ingresados por el usuario son
44    mayores que 0
45    if radio>0 and altura_cilindro > 0:
46        #Se realiza la operación para encontrar el área del
47        cilindro
48        area_cilindro = 2*math.pi*radio*(radio+altura_cilindro)
49    #Se imprime la solución
50    print("El área del cilindro es:",area_cilindro,"u^2")
51    #Si es que el radio y la altura del cilindro son menores
52    que 0
53    else:
```

```
44      #Se le notifica al usuario que el cilindro no existe
45      print("El cilindro no existe")
46      #Se le notifica al usuario que lo intente de nuevo
47      print("Inténtelo de nuevo!")
48      #Se vuelve a empezar
49      cilindro()
```

Programacion

Casos de Prueba y Ejecución del programa La ejecución cuando el cilindro existe es:

```
Ingrese el radio:5
Ingrese la altura:4
El área del cilindro es: 282.7433388230814 u^2
```

Fig. 13. Ejecución Cilindro

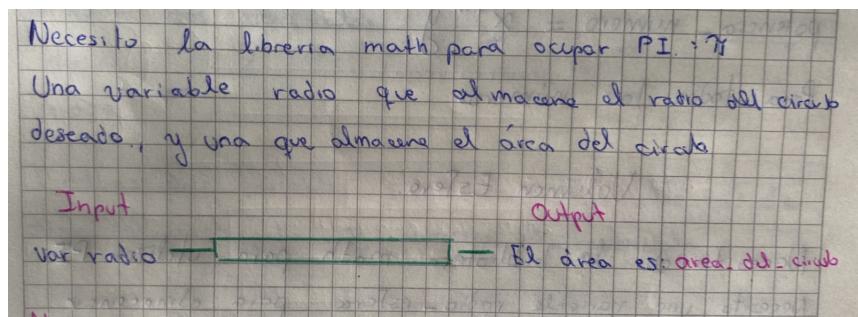
Cuando un cilindro no existe:

```
Ingrese un radio para calcular el área:asd
Ingrese un número válido
Ingrese un radio para calcular el área:-789
El círculo no existe
Inténtelo de nuevo!
```

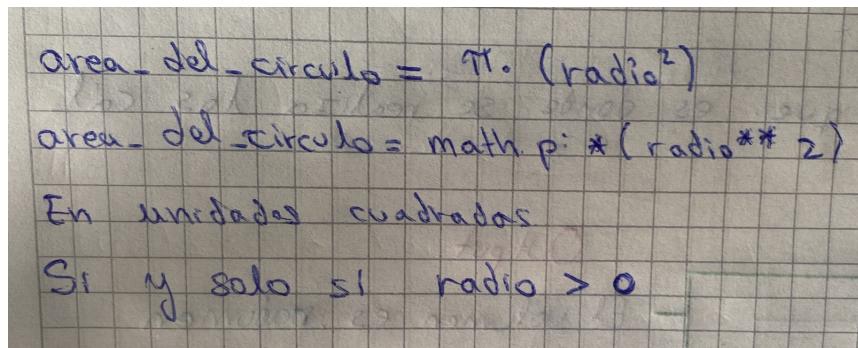
Fig. 14. Ejecución Cilindro inexistente

3.5 Area de un Círculo

Análisis El análisis es el siguiente:

**Fig. 15.** Análisis Área de un Círculo

Modelo El Modelo a continuación:

**Fig. 16.** Modelo Área de un Círculo

Programación Siguiendo el modelo y el análisis:

```

1 def area_circulo():
2     """
3         Función para encontrar el área de un círculo
4     """
5     Parámetros:
6     radio: Dato float ingresado por el usuario
7
8     Retorna:
9     area_del_círculo: Valor en unidades cuadradas
10    correspondiente al área de un círculo
11    """
12    #Se inicia un bucle
13    while True:

```

```

13     #Se emplea la sentencia try/except para validar datos
14     try:
15         #El usuario ingresa un dato tipo float que se
16         #almacena en la variable radio
17         radio = float(input("Ingrese un radio para
18         calcular el área:"))
19         #break
20         break
21     #Si es que el usuario ingresa cualquier otro tipo de
22     #dato que no sea float
23     except ValueError:
24         #Se le notifica al usuario que ingreso un dato
25         #incorrecto
26         print("Ingrese un número válido")
27     #Si el radio es mayor que 0 el área del círculo existe
28     if radio>0:
29         #Cálculo del área del círculo
30         area_del_circulo = math.pi*(radio**2)
31         #Se imprime la solución
32         print("El área del círculo es:",area_del_circulo,"u^2")
33     )
34     #Si el radio es menor que 0, el área no existe
35     else:
36         #Se le notifica al usuario que el círculo no existe
37         print("El círculo no existe")
38         #Se le notifica al usuario que lo intente de nuevo
39         print("Inténtelo de nuevo!")
40         #Se empieza desde el comienzo
41         area_circulo()

```

Programacion

Casos de Prueba y Ejecución del programa Cuando el círculo existe:

```

Ingrese un radio para calcular el área:8.7
El área del círculo es: 237.7871479502114 u^2

```

Fig. 17. Ejecución Área de un Círculo

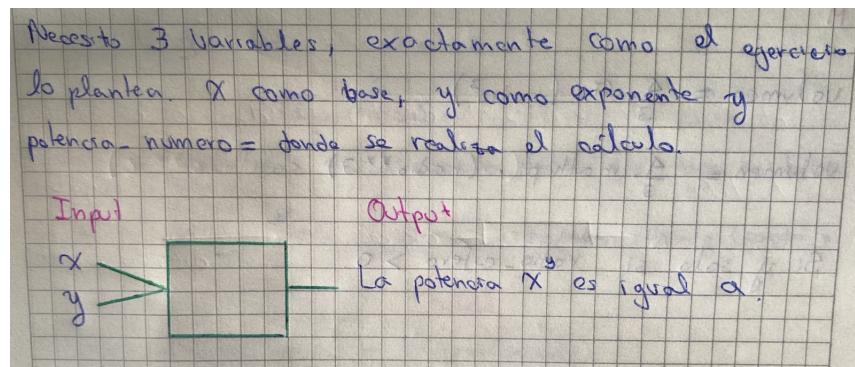
Cuando el círculo no existe:

```
Ingrese un radio para calcular el área:asd
Ingrese un número válido
Ingrese un radio para calcular el área:-789
El círculo no existe
Inténtelo de nuevo!
```

Fig. 18. El Círculo no existe

3.6 Cálculo de Potencias

Análisis El Análisis es el siguiente:

**Fig. 19.** Análisis Potencia

Modelo El Modelo a Continuación:

$$\text{potencia_numero} = x^{**} y.$$

Para todos los números

Fig. 20. Modelo Potencia

Programación Siguiendo el análisis y el modelo:

```
1 def potencia():
2     """
3         Función que sirve para calcular la potencia de cualquier número
4
5     Parámetros:
6         x: Base de una potencia
7         y: Exponente de una potencia
8
9     Retorna:
10        potencia_numero: Potencia x^y
11        """
12
13    #Se inicia un bucle para la variable x
14    while True:
15        #Sentencia try/except para validar datos
16        try:
17            #El usuario ingresa un valor float que se almacena en la variable x
18            x=float(input("Ingrese una base x:"))
19            #break
20            break
21        #Si es que el usuario no ingresó un número float
22        except ValueError:
23            #Se le notifica al usuario que se equivocó
24            print("Ingrese un dato válido!")
25    #Se inicia un bucle para la variable y
26    while True:
27        #Sentencia try/except para validar datos
28        try:
29            #El usuario ingresa un valor float que se almacena en la variable y
30            y=float(input("Ingrese un exponente y:"))
31            #break
32            break
33        #Si el usuario no ingresó un valor float
34        except ValueError:
35            #Se le notifica al usuario que ingresó un dato inválido
36            print("Ingrese un dato válido!")
37    #Se calcula la potencia del número
38    potencia_numero = x**y
39    #Se imprime la solución
40    print("La potencia x^y es igual a:",potencia_numero)
```

Programacion

Casos de Prueba y Ejecución del programa La ejecución del cálculo de potencias es la siguiente:

```
Ingrese una base x:2
Ingrese un exponente y:8
La potencia x^y es igual a: 256.0
```

Fig. 21. Ejecución Cálculo de Potencias

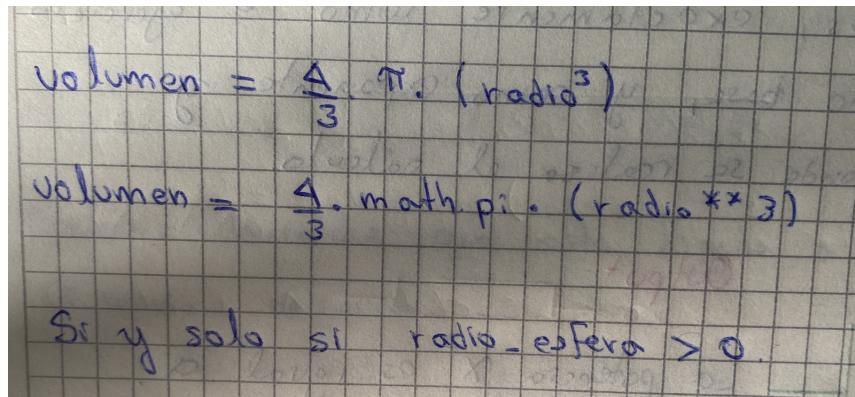
3.7 Volumen de una Esfera

Análisis El Análisis es el siguiente:

Necesito de la biblioteca math para utilizar PI
Necesito una variable radio_esfera para almacenar
el radio
Necesito volumen que es donde se realiza los cálculos
Input Output
radio_esfera —————— El volumen es volumen

Fig. 22. Análisis Volumen de una esfera

Modelo El modelo a continuación:

**Fig. 23.** Modelo Volumen de una esfera

Programación Siguiendo el Análisis y el modelo:

```

1 def volumen_esfera():
2     """
3         Función que permite calcular el volúmen de una esfera por
4         medio de un radio ingresado por el usuario
5
6     Parámetros:
7         radio_esfera: Datos float ingresado por el usuario
8
9     Retorna:
10        volumen: Volumen de la esfera
11        """
12
13    #Bucle True para la variable radio_esfera
14    while True:
15        #Sentencia try/except para validación de datos
16        try:
17            #El usuario ingresa un dato float y se almacena en
18            #la variable radio_esfera
19            radio_esfera=float(input("Ingrese el radio para
20            calcular el volúmen de una esfera:"))
21            #break
22            break
23        #Si el usuario no ingresó un tipo de dato float
24        except ValueError:
25            #Se le notifica al usuario que se equivocó de dato
26            print("Ingrese un dato válido!")
#Si la esfera existe el radio es mayor a 0
if radio_esfera>0:
    #Se calcula el volúmen
    volumen=4/3*math.pi*(radio_esfera**3)

```

```

27      #Se imprime la solución
28      print("El volumen de la esfera es:",volumen,"u^3")
29  #Si el radio es menor a 0 la esfera no existe
30 else:
31     #Se le notifica al usuario que la esfera no existe
32     print("La esfera no existe")
33     #Se le notifica al usuario que lo intente de nuevo
34     print("Inténtelo de nuevo!")
35     #Se repite el proceso
36     volumen_esfera()

```

Programacion

Casos de Prueba y Ejecución del programa Cuando el usuario ingresa los datos correctamente y la esfera existe:

```

Ingrese el radio para calcular el volúmen de una esfera:7.45
El volumen de la esfera es: 1732.038046141617 u^3

```

Fig. 24. Ejecución Volúmen de una esfera

Cuando el usuario ingresa un número negativo y la esfera no existe:

```

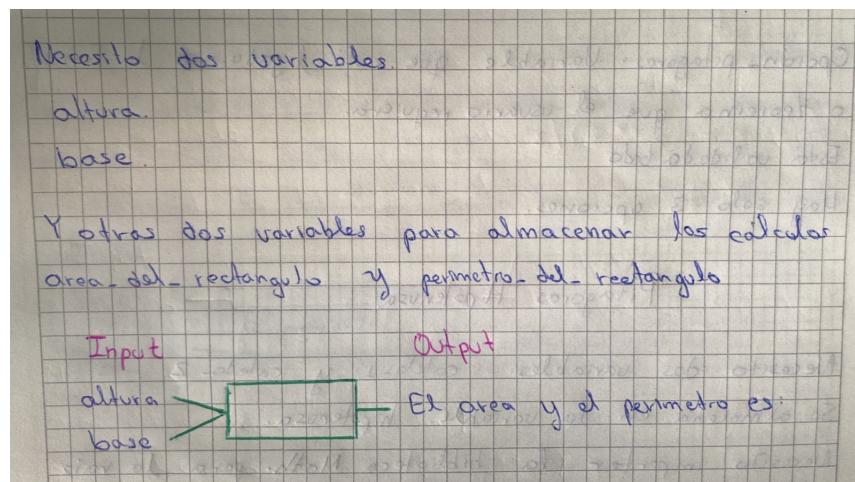
Ingrese el radio para calcular el volúmen de una esfera:-7
La esfera no existe
Inténtelo de nuevo!

```

Fig. 25. Volúmen de esfera inválido

3.8 Área de un Rectángulo

Análisis El análisis es el siguiente:

**Fig. 26.** Análisis Área y Perímetro de un Rectángulo

Modelo El modelo a continuación:

$$\text{área_del_rectángulo} = \text{altura} \times \text{base}$$

$$\text{perímetro_del_rectángulo} = 2\text{altura} + 2\text{base}$$

Si y solo si $\text{base} \wedge \text{altura} > 0$

Fig. 27. Modelo Área y Perímetro de un Rectángulo

Programación Siguiendo el Análisis y el Modelo:

```

1 def area_perimetro_rectangulo():
2     """
3         Se Calcula el área y perímetro de un rectángulo
4
5     Parámetros:
6         altura: Altura del rectángulo ingresado por el usuario
7         base: Base del rectángulo ingresado por el usuario
8
9     Retorna:
10        area_del_rectangulo: Calculando BasexAltura

```

```
11     perimetro_del_rectangulo: Calculando 2Base+2Altura
12     """
13     #Bucle para la variable altura
14     while True:
15         #Sentencia try/except para validar datos
16         try:
17             #El usuario ingresa un dato float y se almacena en
18             #la variable altura
19             altura=float(input("Ingrese la altura:"))
20             #break
21             break
22         #Si el usuario no ingresó un valor float
23         except ValueError:
24             #Se le notifica al usuario que se equivocó
25             print("Ingrese un dato válido!")
26     #Bucle para la variable base
27     while True:
28         #Sentencia try/except para validar datos
29         try:
30             #El usuario ingresa un dato float y se almacena en
31             #la variable base
32             base=float(input("Ingrese la base:"))
33             #break
34             break
35         #Si el usuario ingresó un dato inválido
36         except ValueError:
37             #Se le notifica al usuario que se equivocó
38             print("Ingrese un dato válido!")
39     #Si el rectángulo existe la base y la altura deben ser
40     #mayores a 0
41     if base >0 and altura >0:
42         #Cálculo área del rectángulo
43         area_del_rectangulo=base*altura
44         #Se imprime la solución del área del rectángulo
45         print("El área del rectángulo es:",area_del_rectangulo
46             ," $^2$ ")
47         #Cálculo perímetro del rectángulo
48         perimetro_del_rectangulo=base+base+altura+altura
49         #Se imprime la solución del perímetro del rectángulo
50         print("El perímetro del rectángulo es:",
51             perimetro_del_rectangulo," $^u$ ")
52     #Si el rectángulo no existe
53     else:
54         #Se le notifica al usuario que el rectángulo no existe
55         print("El rectángulo no existe")
56         #Se le anima al usuario a intentarlo de nuevo
57         print("Inténtelo de nuevo!")
58         #Se vuelve a empezar
59         area_perimetro_rectangulo()
```

Programacion

Casos de Prueba y Ejecución del programa La ejecución del programa a continuación, con sus respectivas validaciones:

```
Ingrese la altura:8
Ingrese la base:7
El área del rectángulo es: 56.0 u^2
El perímetro del rectángulo es: 30.0 u
```

Fig. 28. Ejecución Válida

Cuando el usuario ingresa números negativos, o valores inválidos, se le comunica:

```
Ingrese la altura:-7
Ingrese la base:5
El rectángulo no existe
Inténtelo de nuevo!
```

Fig. 29. Ejecución Inválida

3.9 Teorema de Pitágoras - Menú

Análisis El Análisis es el siguiente:

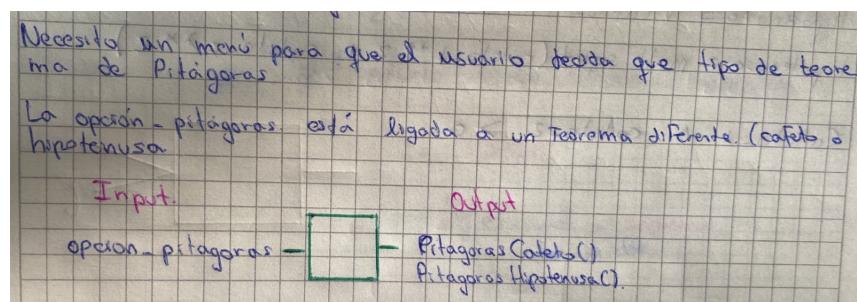


Fig. 30. Análisis Teorema Pitágoras - Menú

Modelo El modelo a continuación:

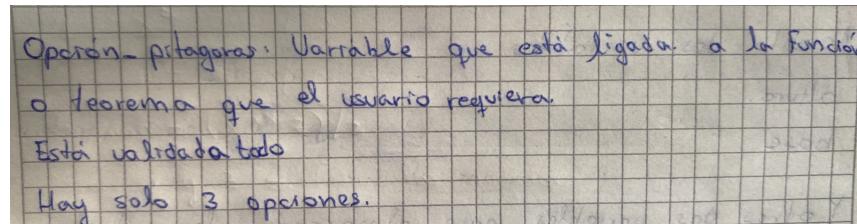


Fig. 31. Modelo Teorema Pitágoras - Menú

Programación Aquí se encuentra un menú para que el usuario elija qué variante del Teorema de Pitágoras necesita:

```

1 def pitagoras_menu():
2     """
3         Menu para que el usuario elija qué tipo de cálculo Pitagó
4         rico requiere
5     """
6     Parámetros:
7     opción_pitagoras: Dato ingresado según requiera el usuario
8
9     Retorna:
10    pitagoras_hipotenusa
11    pitagoras_cateto
12    """
13    #Se inicializa la variable opcion_pitagoras
14    opcion_pitagoras=""
15    #Bucle, mientras la opcion escogida sea diferente de
16    #cuatro
17    while opcion_pitagoras != "4":
18        #Se imprime el menu
19        pitagoras_menu= """
20        MENU PITAGORAS
21
22        1. Calcular Hipotenusa
23        2. Calcular Cateto Faltante
24        3. Volver al menú principal
25
26        Elija su opcion:
27        """
28        #El usuario ingresa la opción que desea
29        opcion_pitagoras = input(pitagoras_menu)
30        #Si la opción elegida es 1

```

```

29      if opcion_pitagoras=="1":
30          #Se llama a la función para calcular la hipotenusa
31          pitagoras_hipotenusa()
32          #break
33          break
34      #Si la opción elegida es 2
35      if opcion_pitagoras=="2":
36          #Se llama a la función para calcular el cateto
            faltante
37          pitagoras_cateto()
38          #break
39          break
40      #Si la opción elegida es 3
41      if opcion_pitagoras=="3":
42          #Se vuelve al menú principal
43          menu()

```

Programacion

Casos de Prueba y Ejecución del programa A continuación el menú de Pitágoras:

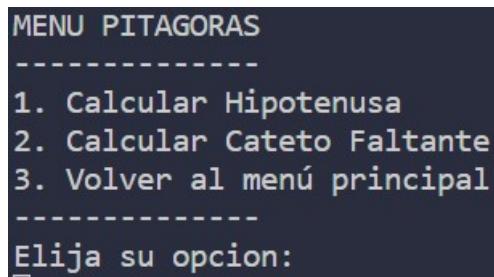
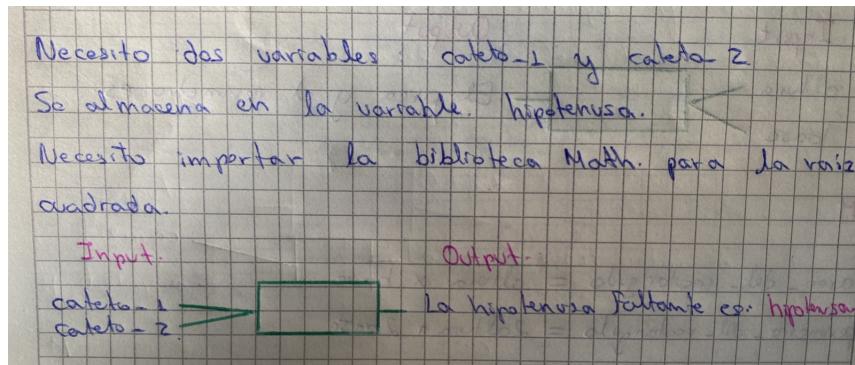


Fig. 32. Menú Pitágoras

3.10 Teorema de Pitágoras - Hipotenusa

Análisis El Análisis es el siguiente:

**Fig. 33.** Análisis Teorema de Pitágoras - Hipotenusa

Modelo El modelo a continuación:

$$\text{hipotenusa} = \sqrt{\text{cateto_1}^2 + \text{cateto_2}^2}$$

$$\text{hipotenusa} = \text{math.sqrt}(\text{cateto_1}^{** 2} + \text{cateto_2}^{** 2})$$

$$\text{Cateto_1} \wedge \text{cateto_2} > 0$$

Fig. 34. Modelo Teorema de Pitágoras - Hipotenusa

Programación Siguiendo el análisis y el modelo:

```

1 def pitagoras_hipotenusa():
2     """
3         Función que calcula la hipotenusa en un triángulo rectángulo
4
5     Parámetros:
6         cateto_1: Cateto 1 ingresado por el usuario
7         cateto_2: Cateto 2 ingresado por el usuario
8
9     Retorna:
10        hipotenusa: Se calcula la hipotenusa
11        """
12    #Bucle para la variable cateto_1
13    while True:
14        #Sentencia try/except para validar datos

```

```

15     try:
16         #El usuario ingresa un dato float y se almacena en
17         #la variable cateto_1
18         cateto_1=float(input("Ingrese el valor de un
19         cateto:"))
20         #break
21         break
22     #El usuario no ingreso un dato correcto
23     except ValueError:
24         #Se le notifica al usuario que se equivocó
25         print("Ingrese un dato válido!")
26 #Bucle para la variable cateto_2
27 while True:
28     #Sentencia try/except para validar datos
29     try:
30         #El usuario ingresa un dato float y se almacena en
31         #la variable cateto_2
32         cateto_2=float(input("Ingrese el valor del otro
33         cateto:"))
34         #break
35         break
36     #El usuario no ingresó un dato correcto
37     except ValueError:
38         #Se le notifica al usuario que se equivocó
39         print("Ingrese un dato válido!")
40 #Para que el rectángulo exista el cateto 1 y el cateto 2
41 #deben ser mayores a 0
42 if cateto_1 >0 and cateto_2 > 0:
43     #Cálculo de la hipotenusa
44     hipotenusa=math.sqrt(cateto_1**2+cateto_2**2)
45     #Solución de la hipotenusa
46     print("La hipotenusa es:",hipotenusa)
47     #Si el triángulo no existe
48 else:
49     #Se le notifica al usuario que el triángulo no existe
50     print("El triángulo no existe")
51     #Se le notifica al usuario que lo intente de nuevo
52     print("Inténtelo de nuevo!")
53     #Se empieza desde el comienzo
54     pitagoras_hipotenusa()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución del cálculo de la Hipotenusa obedeciendo el Teorema de Pitágoras es la siguiente:

```
Ingrese el valor de un cateto:5
Ingrese el valor de la hipotenusa:12
El cateto faltante es: 10.908712114635714
```

Fig. 35. Ejecución Cálculo de la Hipotenusa Válido

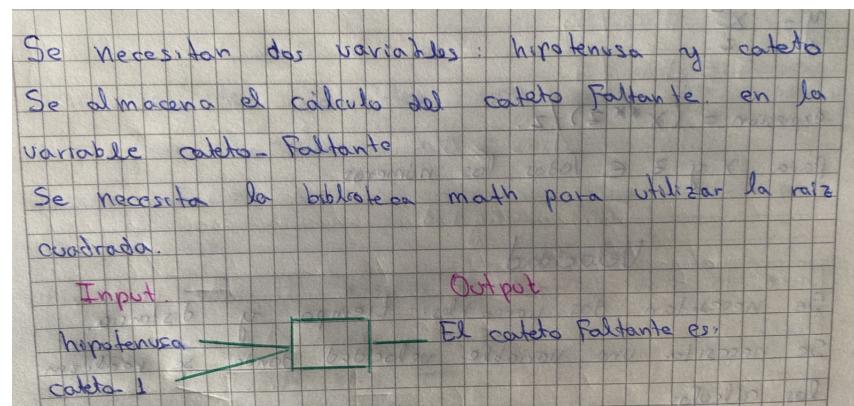
Cuando el usuario ingresa un valor inválido, se le comunica que cometió un error:

```
Ingrese el valor de un cateto:-4
Ingrese el valor de la hipotenusa:45
El triángulo no existe
Inténtelo de nuevo!
```

Fig. 36. Ejecución Cálculo de la Hipotenusa inválido

3.11 Teorema de Pitágoras - Cateto

Análisis El Análisis es el siguiente:

**Fig. 37.** Análisis Teorema de Pitágoras - Cateto

Modelo El Modelo a continuación:

17

$$\text{cateto_faltante} = \sqrt{\text{hipotenusa}^2 - \text{cateto_1}^2}$$

$$\text{cateto_faltante} = \text{math.sqrt}(\text{hipotenusa}^{**2} - \text{cateto_1}^{**2})$$

Hipotenusa > Cateto > 0.

Fig. 38. Modelo Teorema de Pitágoras - Cateto

Programación Siguiendo el análisis y el modelo:

```

1 def pitagoras_cateto():
2     """
3         Función para calcular el cateto faltante en un triángulo
4         rectángulo
5     -----
6         Parámetros:
7             cateto: Cateto ingresado por el usuario
8             hipotenusa: Hipotenusa ingresado por el usuario
9
10        Retorna:
11            cateto_faltante: Cateto que hace falta
12        """
13
14    #Bucle para la variable cateto
15    while True:
16        #sentencia try/except para validar datos
17        try:
18            #El usuario ingresa un dato float que se almacena
19            #en la variable cateto
20            cateto=float(input("Ingrese el valor de un cateto:"))
21        except ValueError:
22            #Se le notifica al usuario que ingresó un dato inv
23            #álido
24            print("Ingrese un dato válido!")
25    #Bucle para la variable hipotenusa
26    while True:
27        #Sentencia try/except para validar datos
28        try:
29            #El usuario ingresa un dato float que se almacena
            en la variable hipotenusa
            hipotenusa=float(input("Ingrese el valor de la
            hipotenusa:"))

```

```

30         #break
31         break
32     #Si el usuario ingresó un dato incorrecto
33     except ValueError:
34         #Se le notifica al usuario que se equivocó
35         print("Ingrese un valor válido!")
36     #Si el triángulo existe la hipotenusa y el cateto es mayor
37     #que 0
38     if hipotenusa >0 and cateto >0:
39         #Cálculo del cateto faltante
40         cateto_faltante=math.sqrt(hipotenusa**2-cateto**2)
41         #Se imprime la solución
42         print("El cateto faltante es:",cateto_faltante)
43     #Si el triángulo no existe
44     else:
45         #Se le notifica al usuario que el triángulo no existe
46         print("El triángulo no existe")
47         #Se le notifica al usuario que lo intente de nuevo
48         print("Inténtelo de nuevo!")
49         #se empieza desde el comienzo
50         pitagoras_cateto()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución del cateto válido es la siguiente:

```

Ingrese el valor de un cateto:2
Ingrese el valor del otro cateto:2
La hipotenusa es: 2.8284271247461903

```

Fig. 39. Ejecución Cateto Inválido

Cuando el usuario ingresa valores incorrectos se le comunica:

```

Ingrese el valor de un cateto:-5
Ingrese el valor del otro cateto:*78
Ingrese un dato válido!
Ingrese el valor del otro cateto:12
El triángulo no existe
Inténtelo de nuevo!

```

Fig. 40. Ejecución Cateto inválido

3.12 Expresión $y=x$ a la z entre 2

Análisis El análisis es el siguiente:

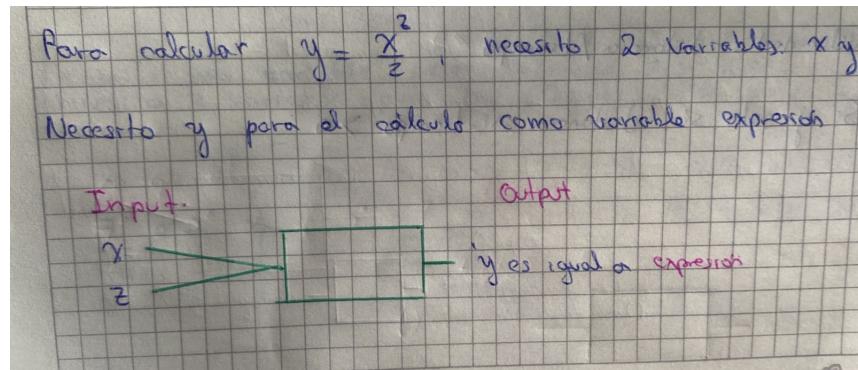


Fig. 41. Análisis de la expresión

Modelo El modelo a continuación:

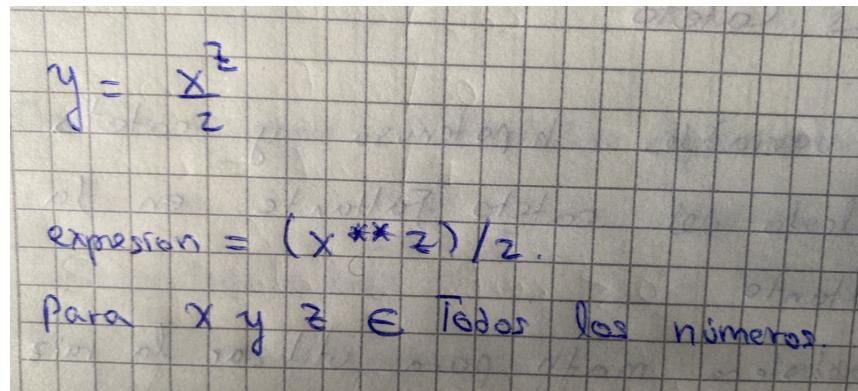


Fig. 42. Modelo de la expresión

Programación Guiándose en el Análisis y Modelo:

```

1 def calculo_expresion():
2     """
3         Función para calcular la expresión  $y=x^z/2$ 
4     """
5     Parámetros:
  
```

```

6     x: Dato float ingresado por el usuario es la base de la
7     potencia
8     z: Dato float ingresado por el usuario es el exponente de
9     la potencia
10    -----
11    Retorna:
12    expresion: Resultado de la expresión
13    """
14    #Bucle para la variable x
15    while True:
16        #Sentencia try/except para validación de datos
17        try:
18            #El usuario ingresa un valor float y se almacena
19            en la variable x
20            x=float(input("Ingrese el valor de x:"))
21            #break
22            break
23        #Si el usuario ingreso un dato inválido
24        except ValueError:
25            #Se le comunica al usuario que se equivocó
26            print("Ingrese un dato válido!")
27    #Bucle para la variable z
28    while True:
29        #Sentencia try/except para validación de datos
30        try:
31            #El usuario ingresa un valor float y se almacena
32            en la variable z
33            z=float(input("Ingrese el valor de z:"))
34            #break
35            break
36        #Si el usuario ingresa un dato inválido
37        except ValueError:
38            #Se le comunica al usuario que se equivocó
39            print("Ingrese un dato válido!")
#Cálculo de la expresión
expresion=(x**z)/2
#Se imprime la solución
print("La expresión es igual a y=",expresion)

```

Programacion

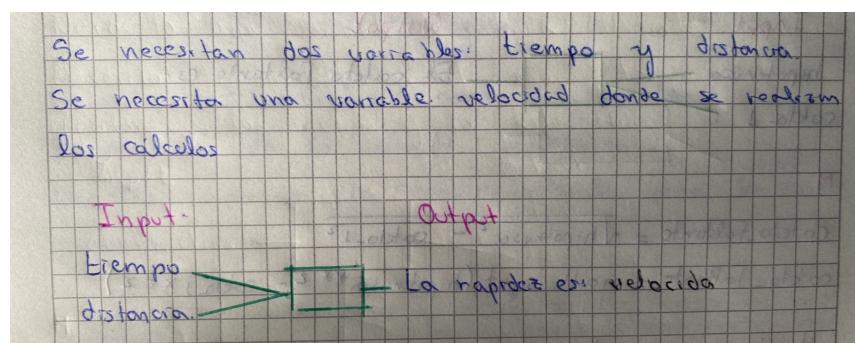
Casos de Prueba y Ejecución del programa A continuación la ejecución del programa:

```
Ingrese el valor de x:54
Ingrese el valor de z:2
La expresión es igual a y= 1458.0
```

Fig. 43. Ejecución $y=x$ a la $z/2$

3.13 Calcular la Rapidez

Análisis El análisis es:

**Fig. 44.** Análisis Velocidad

Modelo El modelo es:

$$\text{Velocidad} = \frac{d}{t}$$

Velocidad = distancia / tiempo
Si y solo si distancia & tiempo > 0

Fig. 45. Modelo Velocidad

Programación Siguiendo el análisis y el modelo:

```
1 def velocidad_fisica():
2 """
3     Función para calcular la rapidez , teniendo el tiempo y la
4     distancia
5     -----
6     Parámetros:
7         tiempo: Dato ingresado por el usuario
8         distancia: Dato ingresado por el usuario
9     -----
10    Retorna:
11        velocidad: Cálculo de la distancia dividida para el tiempo
12        """
13    #Bucle para la variable tiempo
14    while True:
15        #Sentencia try/except para validación de datos
16        try:
17            #El usuario ingresa el tiempo y se almacena en la
18            #variable tiempo
19            tiempo=float(input("Ingrese el tiempo en segundos:"))
20            #break
21            break
22            #Si el usuario ingresó un dato inválido
23            except ValueError:
24                #Se le notifica al usuario que se equivocó
25                print("Ingrese un dato válido!")
26    #Bucle para la variable distancia
27    while True:
28        #Sentencia try/except para validación de datos
29        try:
30            #El usuario ingresa la distancia y se almacena en
31            #la variable distancia
32            distancia=float(input("Ingrese la distancia en
33            metros:"))
34            #break
35            break
36            #Si el usuario ingresó un dato inválido
37            except ValueError:
38                #Se le notifica al usuario que se equivocó
39                print("Ingrese un valor válido")
40    #Si la distancia y el tiempo son mayores a cero la rapidez
41    #existe
42    if distancia > 0 and tiempo >0:
43        #Cálculo de la velocidad
44        velocidad=distancia/tiempo
45        #Se imprime la solución
46        print("La rapidez es igual a:",velocidad,"m/s^2")
47        #Si la rapidez no existe
48        else:
```

```
44      #Se le informa al usuario que no existe la rapidez
45      print("No existen tiempos y/o distancias negativas")
46      #Se le dice al usuario que lo intente de nuevo
47      print("Inténtelo de nuevo!")
48      #Se repite el proceso
49      velocidad_fisica()
```

Programacion

Casos de Prueba y Ejecución del programa A continuación la ejecución del Programa

```
Ingrese el tiempo en segundos:5
Ingrese la distancia en metros:100
La rapidez es igual a: 20.0 m/s^2
```

Fig. 46. Cálculo de la rapidez válida

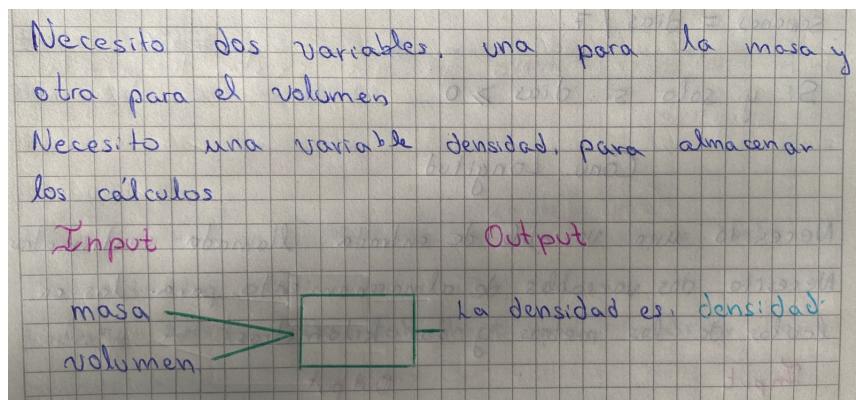
Cuando el usuario ingresa un valor incorrecto se le notifica al usuario:

```
Ingrese el tiempo en segundos:-45
Ingrese la distancia en metros:45
No existen tiempos y/o distancias negativas
Inténtelo de nuevo!
```

Fig. 47. Cálculo de la rapidez inválida

3.14 Densidad de un objeto

Análisis El análisis es el siguiente:

**Fig. 48.** Análisis Densidad

Modelo El modelo a continuación:

$$\text{densidad} = \frac{\text{masa}}{\text{volumen}}$$

$$\text{densidad} = \text{masa} / \text{volumen}$$

Sí y solo si $\text{masa} > 0$ y $\text{volumen} > 0$

Fig. 49. Modelo Densidad

Programación Siguiendo el análisis y el modelo:

```

1 def calculo_densidad():
2     """
3         Función para calcular la densidad de un objeto
4
5     Parámetros:
6         masa: Valor ingresado por el usuario
7         volumen: Valor ingresado por el usuario
8
9     Retorna:

```

```

10  densidad
11 """
12 #Bucle para la variable masa
13 while True:
14     #Sentencia try/except para validar datos
15     try:
16         #El usuario ingresa el dato de la masa
17         masa=float(input("Ingrese la masa del objeto en
kgs:"))
18         #break
19         break
20     #Si el usuario ingresó un dato inválido
21     except ValueError:
22         #Se le notifica al usuario que se equivocó
23         print("Ingrese un dato válido!")
24 #Bucle para la variable volumen
25 while True:
26     #Sentencia try/except para validar datos
27     try:
28         #El usuario ingresa el dato del volúmen
29         volumen=float(input("Ingrese el volumen en m^3:"))
30         #break
31         break
32     #Si el usuario ingresó un dato inválido
33     except ValueError:
34         #Se le notifica al usuario que se equivocó
35         print("Ingrese un dato válido!")
36 #Si la densidad existe
37 if masa>0 and volumen >0:
38     #Cálculo de la densidad
39     densidad=masa/volumen
40     #Se imprime la solución
41     print("La densidad es igual a:",densidad,"kg/m^3")
42 #Si la densidad no existe
43 else:
44     #Se le notifica al usuario que la densidad no existe
45     print("No existen masa y/o volumen menor o igual a 0")
46     #Se le alienta al usuario a empezar de nuevo
47     print("Inténtelo de nuevo!")
48     #Se empieza de nuevo
49     calculo_densidad()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:

```
Ingrese la masa del objeto en kgs:458
Ingrese el volumen en m^3:74
La densidad es igual a: 6.1891891891891895 kg/m^3
```

Fig. 50. Cálculo Densidad Válida

Cuando el usuario ingresa un dato erróneo se le notifica:

```
Ingrese la masa del objeto en kgs:-48
Ingrese el volumen en m^3:67
No existen masa y/o volumen menor o igual a 0
Inténtelo de nuevo!
```

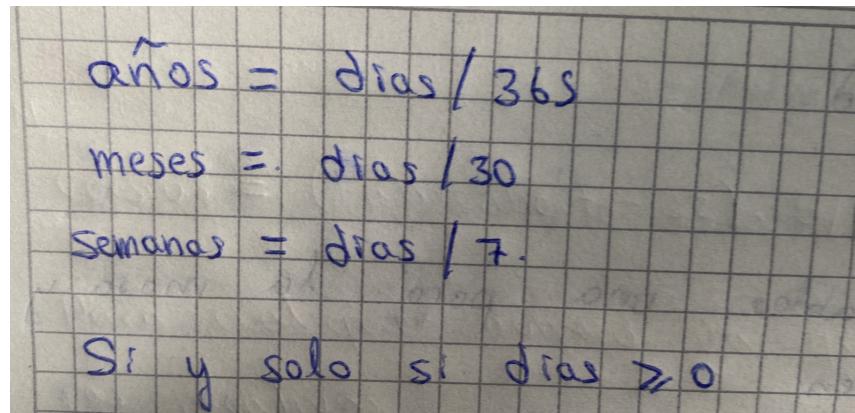
Fig. 51. Cálculo Densidad Inválida

3.15 Conversión de días

Análisis El análisis es el siguiente:

**Fig. 52.** Análisis conversión de días

Modelo El modelo a continuación:

**Fig. 53.** Modelo conversión de días

Programación Sigueindo el análisis y el modelo:

```

1 def conversion_dias():
2     """
3         Función para convertir días en años, meses y semanas
4     """
5     Parámetro:
6     días: Dato ingresado por el usuario
7
8     Retorna:
9     años: días convertidos en años
10    meses: días convertidos en meses
11    semanas: días convertidos en semanas
12    """
13
14    #Bucle para la variable días
15    while True:
16        #Sentencia try/except para validar datos
17        try:
18            #El usuario ingresa el número de días
19            días=int(input("Ingrese el número de días:"))
20            #break
21            break
22        #Si el usuario ingresó un dato equivocado
23        except ValueError:
24            #Se le notifica al usuario que se equivocó
25            print("Ingrese un valor válido")
26    #Si la conversión puede realizarse
27    if días >= 0:
28        #Conversión a años
29        años=días//365
            #Se imprime los días trasnformados en años

```

```

30     print(dias , "días es:",años , "años")
31     #Conversión a meses
32     meses=dias//30
33     #Se imprime los días transformados en meses
34     print(dias , "días son:",meses , "meses")
35     #Conversión a semanas
36     semanas=dias//7
37     #Se imprime los días transformados en semanas
38     print(dias , "días son:",semanas , "semanas")
39     #Si no se puede realizar la conversión
40     else:
41         #Se le notifica al usuario que ingrese un dato válido
42         print("Ingrese un dato válido")
43         #Se alienta al usuario a empezar de nuevo
44         print("Inténtelo de nuevo!")
45         #Se empieza de nuevo
46         conversion_días()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:

```

Ingrese el número de días:952
952 días son: 2 años
952 días son: 31 meses
952 días son: 136 semanas

```

Fig. 54. Conversión de Días Válido

```

Ingrese el número de días:-452
Ingrese un dato válido
Inténtelo de nuevo!

```

Fig. 55. Conversión de Días Inválido]

3.16 Conversión de Longitud

Análisis El Análisis es el siguiente:

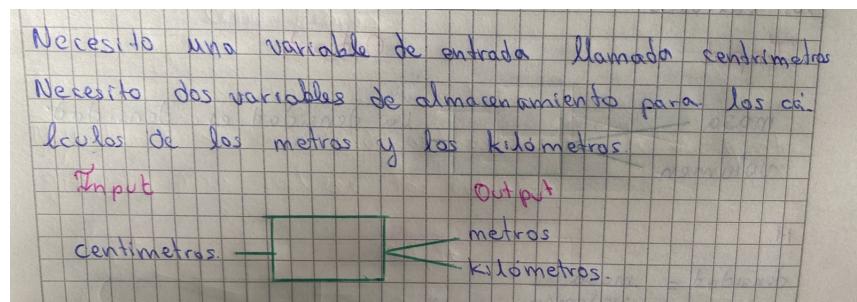


Fig. 56. Análisis Conversión de Longitud

Modelo El Modelo a continuación:

$$\begin{aligned} \text{metros} &= \text{centímetros} / 100 \\ \text{kilómetros} &= \text{centímetros} / 1000 \end{aligned}$$

Fig. 57. Modelo Conversión de Longitud

Programación Siguiendo el Análisis y el Modelo:

```
1 def conversion_longitud():
2     """
3         Función para convertir cm en m y km
4     -----
5         Parámetros:
6             centimetros: Dato ingresado por el usuario
7
8         Retorna:
9             metros: cm transformados en m
10            kilometros: cm transformados en km
11        """
12
13    #Bucle para la variable centímetros
14    while True:
```

```

14     #Sentencia para validación de datos
15     try:
16         #El usuario ingresa el valor de los centímetros
17         centimetros=int(input("Ingrese la longitud en cm:"))
18     )
19         #break
20         break
21     #Si el usuario no ingresó un valor correcto
22     except ValueError:
23         #Se le notifica al usuario que se equivocó
24         print("Ingrese un valor válido")
25     #Si la conversión puede realizarse
26     if centimetros>=0:
27         #Conversión a metros
28         metros=centimetros//100
29         #Se imprime la conversión a metros
30         print(centimetros,"cm son:",metros,"m")
31         #Conversión a kilómetros
32         kilometros=centimetros//1000
33         #Se imprime la conversión a kilómetros
34         print(centimetros,"cm son:",kilometros,"km")
35     #Si la conversión no puede realizarse
36     else:
37         #Se le notifica al usuario que no puede realizarse la
38         #conversión
39         print("Ingrese un valor mayor o igual a 0")
40         #Se alienta al usuario a empezar de 0
41         print("Inténtelo de nuevo!")
42         #Se inicia de nuevo el proceso
43         conversion_longitud()

```

Programacion

Casos de Prueba y Ejecución del programa La Ejecución del programa es la siguiente:

```

Ingrese la longitud en cm:75468
75468 cm son: 754.68 m
75468 cm son: 75.468 km

```

Fig. 58. Conversión de longitudes válido

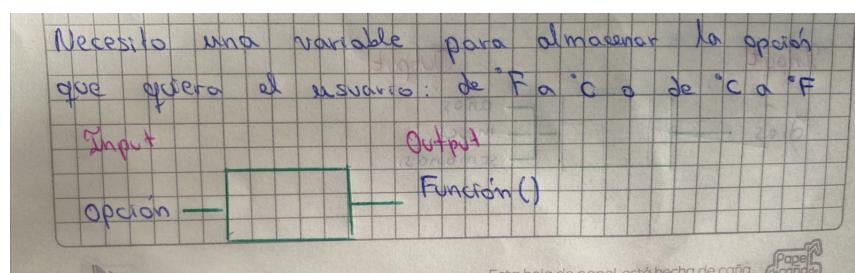
Cuando el usuario ingresa un dato ilógico se le comunica inmediatamente

```
Ingrese la longitud en cm:-785
Ingrese un valor mayor o igual a 0
Inténtelo de nuevo!
```

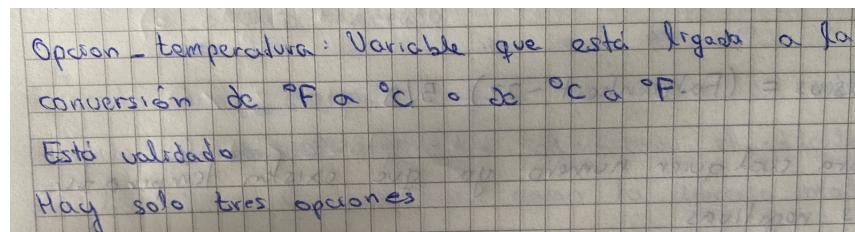
Fig. 59. Conversión de longitudes inválido

3.17 Conversión de Temperatura - Menú

Análisis El Análisis es el siguiente:

**Fig. 60.** Análisis de Conversión de Temperatura - Menú

Modelo El Modelo a continuación:

**Fig. 61.** Modelo de Conversión de Temperatura - Menú

Programación Menú para que el usuario seleccione que tipo de transformación de Temperatura desea:

```

1 def menu_temperatura():
2     """
3         Función de menú para que el usuario escoja que
4             transformación quiere realizar
    
```

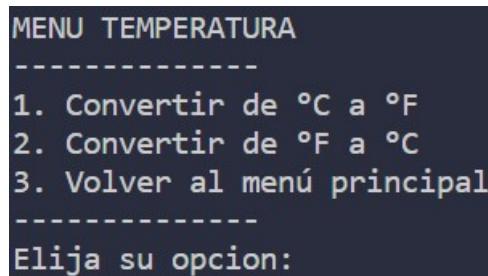
```

5  Parámetros:
6  opcion_temperatura: Opción que escoja el usuario según
7  requiera
8
9  Retorna:
10 conversion_celsius
11 conversion_fahrenheit
12 """
13 #Se inicializa la variable opcion_temperatura
14 opcion_temperatura = ""
15 #Bucle mientras la opción no sea 4
16 while opcion_temperatura != "4":
17     #Se imprime el menú
18     menu_temperatura = """
19         MENU TEMPERATURA
20
21             1. Convertir de C a F
22             2. Convertir de F a C
23             3. Volver al menú principal
24
25     Elija su opción:
26 """
27     #El usuario ingresa la conversión que requiera
28     opcion_temperatura = input(menu_temperatura)
29     #Si escogió 1
30     if opcion_temperatura == "1":
31         #Se llama a la función conversión a celcius
32         conversion_celsius()
33         break
34     #Si escogió la opción 2
35     if opcion_temperatura == "2":
36         #Se llama a la función conversión a fahrenheit
37         conversion_fahrenheit()
38         break
39     #Si escogió la opción 3
40     if opcion_temperatura == "3":
41         #Se retorna al menú principal
42         menu()
43

```

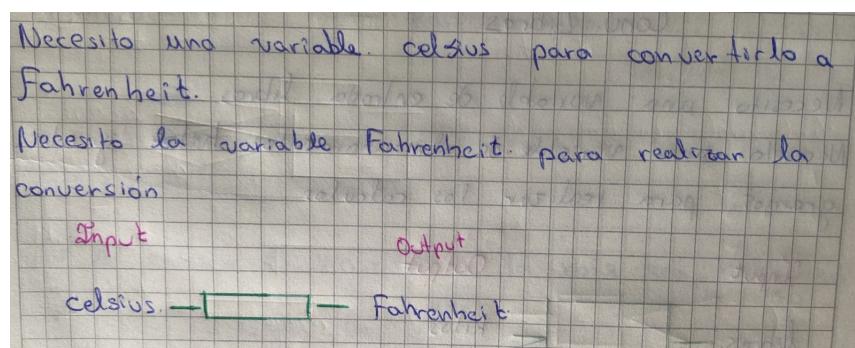
Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:

**Fig. 62.** Menú Conversión de Temperatura

3.18 Conversión de Temperatura - De Celsius a Fahrenheit

Análisis El análisis a continuación:

**Fig. 63.** Análisis Conversión de Temperatura de Celsius a Fahrenheit

Modelo El modelo es el siguiente:

$$\text{Fahrenheit} = (\text{celsius} * \frac{9}{5}) + 32$$

Para cualquier número, ya que existen temperaturas negativas.

Fig. 64. Modelo Conversión de Temperatura de Celsius a Fahrenheit

Programación Siguiendo el Análisis y el modelo:

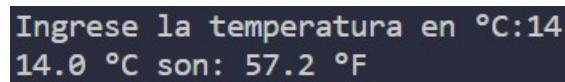
```

1 def conversion_celcius():
2 """
3     Función para convertir C a F
4
5     Parámetros:
6         celsius: Dato ingresado por el usuario
7
8     Retorna:
9         fahrenheit: C convertidos a F
10    """
11
12    #Bucle para la variable celsius
13    while True:
14        #Sentencia try/except para validación de datos
15        try:
16            #El usuario ingresa los grados celsius
17            celsius=float(input("Ingrese la temperatura en C
18 :"))
19            #break
20            break
21        #Si el usuario ingresó un dato inválido
22        except ValueError:
23            #Se le notifica al usuario que ingresó un dato inv
24            #álido
25            print("Ingrese un dato correcto!")
26    #Conversión a F
27    fahrenheit=(celsius*9/5)+32
28    #Imprime la conversión
29    print(celsius," °C son:",fahrenheit," °F ")

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución de Celsius a Fahrenheit es la siguiente:



```
Ingrese la temperatura en °C:14
14.0 °C son: 57.2 °F
```

Fig. 65. Conversión de Celsius a Fahrenheit

3.19 Conversión de Temperatura - De Fahrenheit a Celsius

Análisis El Análisis es:

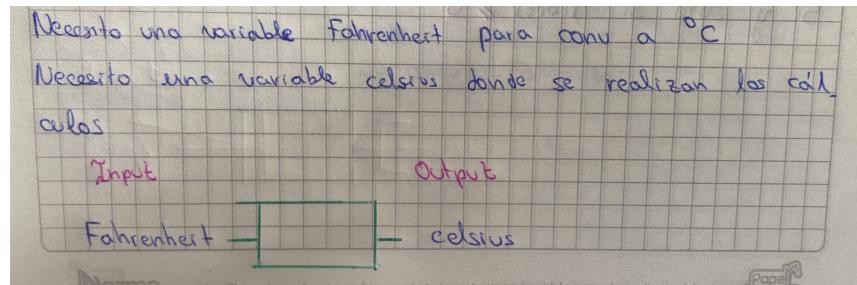


Fig. 66. Análisis Conversión de Temperatura de Fahrenheit a Celcius

Modelo El Modelo es:

$$\text{Celsius} = (\text{Fahrenheit} - 32) \cdot \frac{5}{9}$$

Fig. 67. Modelo Conversión de Temperatura de Fahrenheit a Celcius

Programación Siguiendo el Análisis y el modelo:

```
1 def conversion_fahrenheit():
2     """
3         Función para convertir F a C
4     -----
5         Parámetros:
6             fahrenheit: Dato ingresado por el usuario
7
8         Retorna:
9             celcius: Conversión de F a C
10        """
11    #Bucle para la variable fahrenheit
12    while True:
13        #Sentencia try/except para validación de datos
14        try:
15            #El usuario ingresa los F
16            fahrenheit=float(input("Ingrese la temperatura en F :"))
17            #break
18            break
```

```

19     #Si el usuario no ingresó un dato válido
20     except ValueError:
21         #Se le notifica al usuario que ingresó un dato
22         erroneo
23         print("Ingrese un dato válido!")
24     #Conversión de F a C
25     celcius=(fahrenheit -32)*5/9
26     #Se imprime la conversión a C
27     print(fahrenheit , " F son:",celcius , " C ")

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución de Fahrenheit a Celsius es la siguiente:

```
Ingrese la temperatura en °F:75
75.0 °F son: 23.8888888888889 °C
```

Fig. 68. Conversión de Fahrenheit a Celcius

3.20 Conversión de Masa

Análisis El análisis a continuación:

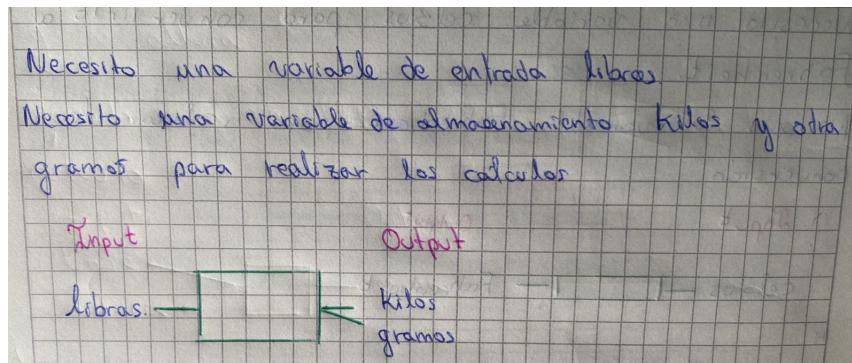


Fig. 69. Análisis Conversión de Masa

Modelo El Modelo es el siguiente:

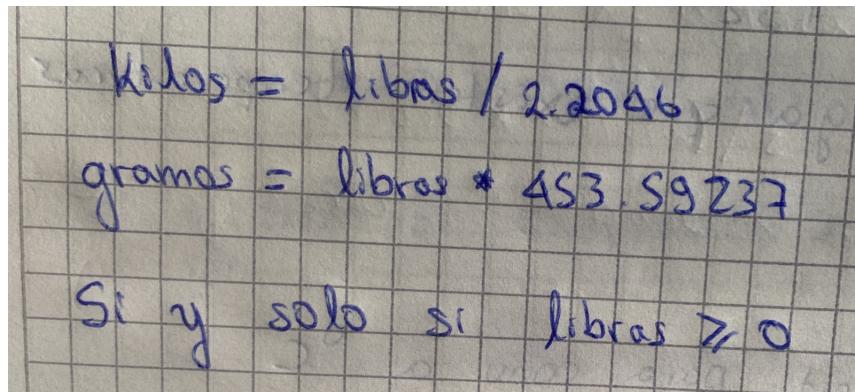


Fig. 70. Modelo Conversión de Masa

Programación Siguiendo el análisis y el modelo:

```
1 def convertir_libras():
2     """
3         Función para convertir libras a kilos y gramos
4     -----
5     Parámetros:
6         libras: Dato ingresado por el usuario
7     -----
8     Retorna:
9         kilos: Conversión de lbs a kg
10        gramos: Conversión de lbs a g
11    """
12
13 #Bucle para la variable libras
14 while True:
15     #Sentencia try/except para validación de datos
16     try:
17         #El usuario ingresa el número de libras
18         libras=float(input("Ingrese el valor en libras:"))
19         #break
20         break
21     #Si el usuario ingresó un dato inválido
22     except ValueError:
23         #Se le notifica al usuario que se equivocó
24         print("Ingrese un dato válido!")
25 #Si la conversión de libras es posible realizarse
26 if libras >=0:
27     #Conversión de libras a kilos
28     kilos=libras/2.2046
29     #Se imprime la conversion de libras a kilos
30     print(libras,"lbs son:",kilos,"kg")
31     #Conversión de libras a gramos
```

```

31     gramos=libras*453.59237
32     #Se imprime la conversión de libras a gramos
33     print(libras,"lbs son:",gramos,"g")
34     #Si la conversión no puede realizarse
35     else:
36         #Se le notifica al usuario que ingrese un valor igual
37         o mayor a 0
38         print("Ingrese un valor igual o mayor a 0")
39         #Se anima al usuario a intentarlo de nuevo
40         print("Inténtelo de nuevo!")
41         #Se empieza desde el comienzo
        convertir_libras()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:
Cuando el usuario ingresa un dato ilógico se le comunica:

```

Ingrese el valor en libras:45
45.0 lbs son: 20.411866098158395 kg
45.0 lbs son: 20411.65665 g

```

Fig. 71. Conversión de lbs a kg y g

```

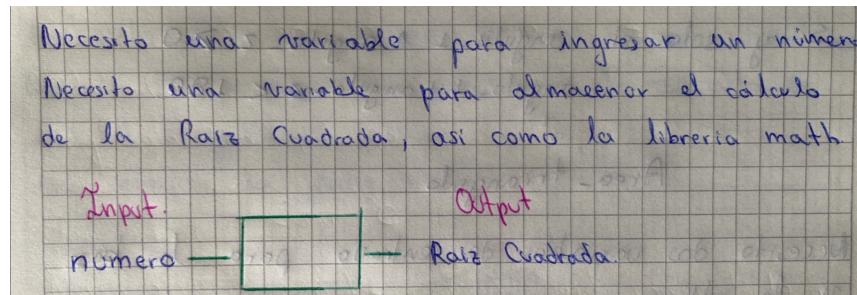
Ingrese el valor en libras:-756
Ingrese un valor igual o mayor a 0
Inténtelo de nuevo!

```

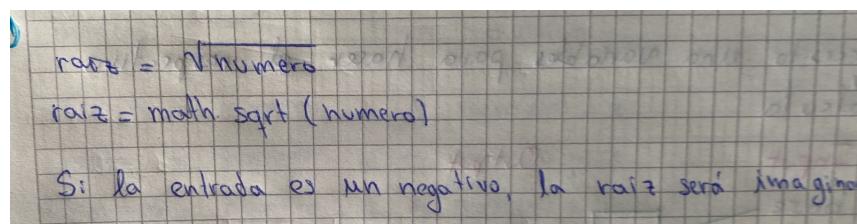
Fig. 72. Conversión de lbs a kg y g Inválido

3.21 Raíz Cuadrada

Análisis El Análisis es el siguiente:

**Fig. 73.** Análisis Raíz Cuadrada

Modelo El modelo a continuación:

**Fig. 74.** Modelo Raíz Cuadrada

Programación Siguiendo el análisis y el modelo:

```

1 def raiz_cuadrada():
2     """
3         Función para encontrar la raíz cuadrada de cualquier número
4     """
5     Parámetros:
6     número: Dato ingresado por el usuario
7
8     Retorna:
9     raiz: Raíz cuadrada del número ingresado
10    """
11    #Bucle para la variable numero
12    while True:
13        #Sentencia try/except para validación de datos
14        try:
15            #El usuario ingresa un número
16            numero=float(input("Ingrese un número:"))
17            #break

```

```

18         break
19     #Si el usuario no ingresó un dato válido
20     except ValueError:
21         #Se le notifica al usuario que se equivocó
22         print("Ingrese un valor válido!")
23     #Si la raíz es de un número natural
24     if numero >=0:
25         #Cálculo de la raíz
26         raiz=math.sqrt(numero)
27         #Se imprime la solución de la raíz
28         print("La raíz cuadrada de:",numero,"es:",raiz)
29     #Si la raíz cuadrada es un número imaginario
30     else:
31         #Cálculo de la raíz
32         raiz2=math.sqrt(numero*(-1))
33         #Se imprime la solución de la raíz imaginaria
34         print("La raíz cuadrada de:",numero,"es:",raiz2 , "i")

```

Programacion

Casos de Prueba y Ejecución del programa La solución de la raíz cuadrada de un número natural es la siguiente:

```

Ingrese un número:64
La raíz cuadrada de: 64.0 es: 8.0

```

Fig. 75. Raíz Cuadrada de un número natural

Cuando se tiene un número negativo, la raíz cuadrada es imaginaria:

```

Ingrese un número:-7548
La raíz cuadrada de: -7548.0 es: 86.87922651589389 i

```

Fig. 76. Raíz Cuadrada de un número negativo

3.22 Ángulo Faltante en un Triángulo

Análisis El análisis es el siguiente:

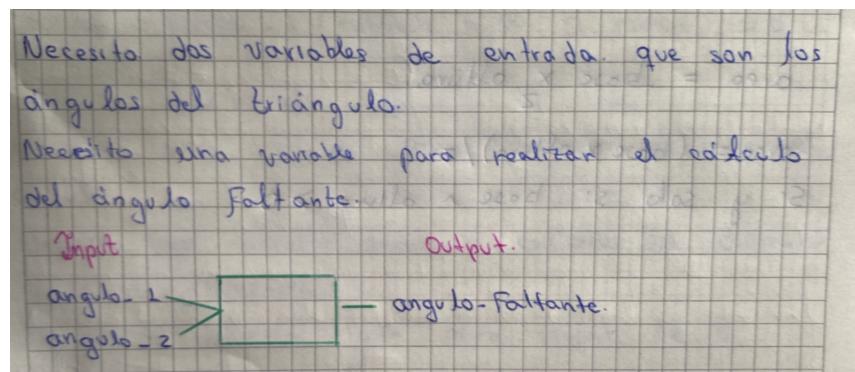


Fig. 77. Análisis Ángulo Faltante en un Triángulo

Modelo El modelo a continuación:

$$\text{angulo-faltante} = 180 - (\text{angulo-1} + \text{angulo-2})$$

Si y solo si $\text{angulo-1} + \text{angulo-2} > 0$
 $\text{angulo-1} + \text{angulo-2} < 180$

Fig. 78. Modelo Ángulo Faltante en un Triángulo

Programación Siguiendo el análisis y el diseño:

```
1 def encontrar_angulo():
2     """
3         Función para encontrar el ángulo restante de un triángulo
4
5     Parámetros:
6         angulo_1: Dato ingresado por el usuario
7         angulo_2: Dato ingresado por el usuario
8
9     Retorna:
10        angulo_faltante: El ángulo que falta del triángulo
11        """
12    #Bucle para la variable angulo_1
13    while True:
14        #Sentencia try/except para validación de datos
15        try:
```

```

16      #El usuario ingresa el ángulo 1
17      angulo_1=float(input("Ingrese el ángulo 1:"))
18      #break
19      break
20      #Si el usuario ingresó un dato inválido
21      except ValueError:
22          #Se le notifica al usuario que se equivocó
23          print("Ingrese un valor válido!")
24      #Bucle para la variable angulo_2
25      while True:
26          #Sentencia try/except para validación de datos
27          try:
28              #El usuario ingresa el ángulo 2
29              angulo_2=float(input("Ingrese el ángulo 2:"))
30              #break
31              break
32              #Si el usuario ingresó in dato inválido
33              except ValueError:
34                  #Se le notifica al usuario que se equivocó
35                  print("Ingrese un valor válido!")
36      #Se verifica que el triángulo exista
37      if angulo_1>0 and angulo_2 > 0 and angulo_1+angulo_2 <180:
38          #Cálculo del ángulo faltante
39          angulo_faltante=180-(angulo_1+angulo_2)
40          #Se imprime el ángulo faltante
41          print("El ángulo faltante es:",angulo_faltante)
42      #Si no existe el triángulo
43      else:
44          #Se le notifica al usuario que el triángulo no existe
45          print("Los ángulos que usted ingresó no corresponden a
46          un triángulo!")
47          #Se le alienta al usuario a intentarlo de nuevo
48          print("Inténtelo de nuevo!")
49          #Se empieza de nuevo desde 0
      encontrar_angulo()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:
Cuando el usuario ingresa datos erroneos, se le notifica:]

```

Ingrese el ángulo 1:45
Ingrese el ángulo 2:78
El ángulo faltante es: 57.0

```

Fig. 79. Encontrar el ángulo faltante de un triángulo

Ingrese el ángulo 1:180
 Ingrese el ángulo 2:21
 Los ángulos que usted ingresó no corresponden a un triángulo!
 Inténtelo de nuevo!

Fig. 80. Encontrar el ángulo faltante de un triángulo inválido

3.23 Área de un triángulo

Análisis El Análisis es:

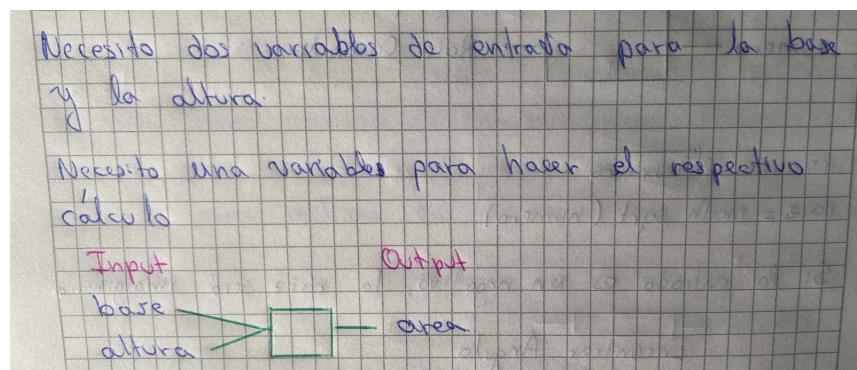


Fig. 81. Análisis Área de un triángulo

Modelo El Modelo es:

$$\text{área} = \frac{\text{base} \times \text{altura}}{2}$$

$$\text{área} = (\text{base} \times \text{altura}) / 2.$$

Si y solo si base & altura > 0.

Fig. 82. Modelo Área de un triángulo

Programación Siguiendo el análisis y el modelo:

```
1 def area_triangulo():
2     """
3         Función para controlar el área de un triángulo
4     -----
5         Parámetros:
6             base_triangulo: Dato ingresado por el usuario
7             altura_triangulo: Dato ingresado por el usuario
8
9         Retorna:
10            area_del_triangulo: Cálculo del área del triángulo
11        """
12
13    #Bucle para la variable base_triangulo
14    while True:
15        #Sentencia try/except para validación de datos
16        try:
17            #El usuario ingresa la base del triángulo
18            base_triangulo=float(input("Ingrese la base de un
19                triángulo:"))
20            #break
21            break
22        #Si el usuario no ingresó un dato válido
23        except ValueError:
24            #Se le notifica al usuario que se equivocó
25            print("Ingrese un dato válido!")
26    #Bucle para la variable altura_triangulo
27    while True:
28        #Sentencia try/except para validación de datos
29        try:
30            #El usuario ingresa la altura del triángulo
31            altura_triangulo=float(input("Ingrese la altura de
32                un triángulo:"))
33            #break
34            break
35        #Si el usuario no ingresó un dato válido
36        except ValueError:
37            #Se le notifica al usuario que se equivocó
38            print("Ingrese un dato válido!")
39    #Si el triángulo existe
40    if base_triangulo>0 and altura_triangulo > 0:
41        #Cálculo del área del triángulo
42        area_del_triangulo=(base_triangulo*altura_triangulo)/2
43        #Se imprime la solución del área del triángulo
44        print("El área del triángulo es:",area_del_triangulo,"
u^2")
45    #Si el triángulo no existe
46    else:
47        #Se le notifica al usuario que el triángulo no existe
```

```
45     print("El triángulo no existe")
46     #Se alienta al usuario a intentarlo de nuevo
47     print("Inténtelo de nuevo!")
48     #Se empieza desde el comienzo
49     area_triangulo()
```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:
Si el usuario ingresa datos ilógicos, se le notifica:

```
Ingrese la base de un triángulo:45
Ingrese la altura de un triángulo:98
El área del triángulo es: 2205.0 u^2
```

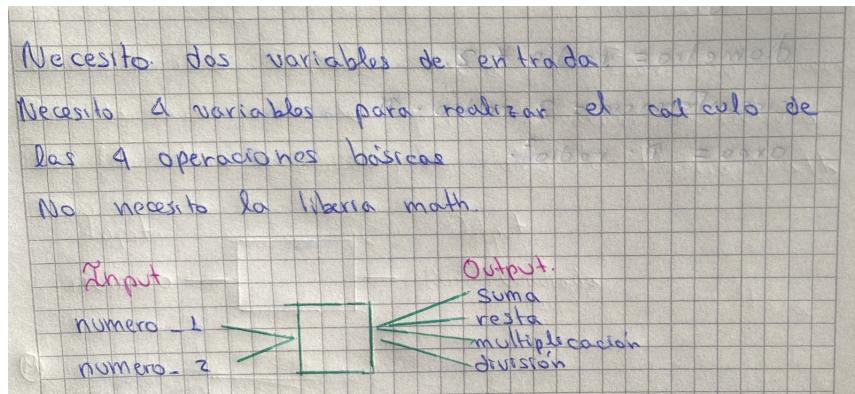
Fig. 83. Área de un triángulo

```
Ingrese la base de un triángulo:-48
Ingrese la altura de un triángulo:-78
El triángulo no existe
Inténtelo de nuevo!
```

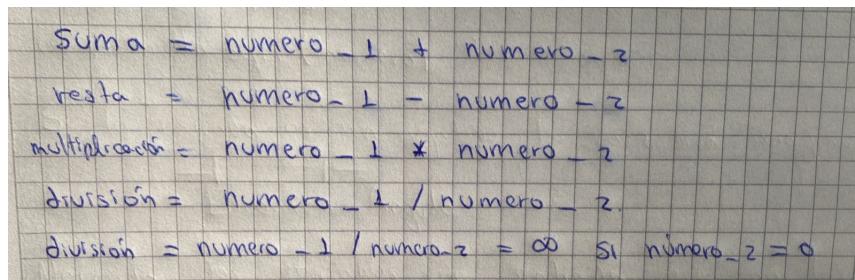
Fig. 84. Área de un triángulo inválido

3.24 Operaciones Básicas de Dos Números

Análisis El análisis del problema es:

**Fig. 85.** Análisis operaciones básicas con dos números

Modelo El modelo del problema es el siguiente:

**Fig. 86.** Modelos operaciones básicas con dos números

Programación Obedeciendo el análisis y el modelo:

```

1 def aritmetica():
2     """
3         Función para calcular las cuatro operaciones básicas
4             mediante dos números
5
5     Parámetros:
6         numero_1: Dato ingresado por el usuario
7         numero_2: Dato ingresado por el usuario
8
    
```

```
9     Retorna:
10    suma
11    resta
12    multiplicación
13    división
14    """
15    #Bucle para la variable numero_1
16    while True:
17        #Sentencia try/except para validación de datos
18        try:
19            #El usuario ingresa el número 1
20            numero_1=float(input("Ingrese el número 1:"))
21            #break
22            break
23            #Si el usuario ingresó un dato inválido
24            except ValueError:
25                #Se le comunica al usuario que ingresó un dato inv
26                álido
27                print("Ingrese un dato inválido!")
28    #Bucle para la variable numero_2
29    while True:
30        #Sentencia try/except para validación de datos
31        try:
32            #El usuario ingresa el número 2
33            numero_2=float(input("Ingrese el número 2:"))
34            #break
35            break
36            #Si el usuario ingresa un dato inválido
37            except ValueError:
38                #Se le comunica al usuario que ingresó un dato inv
39                álido
40                print("Ingrese un dato inválido!")
41    #Se imprimen las soluciones
42    print("Las operaciones básicas con los números ingresados
43    son:")
44    #Cálculo de la suma
45    suma=numero_1+numero_2
46    #Impresión de la suma
47    print(numero_1,"+",numero_2,"=",suma)
48    #Cálculo de la resta
49    resta=numero_1-numero_2
50    #Impresión de la resta
51    print(numero_1,"-",numero_2,"=",resta)
52    #Cálculo de la multiplicación
53    multiplicacion=numero_1*numero_2
54    #Impresión de la multiplicación
55    print(numero_1,"*",numero_2,"=",multiplicacion)
    #División que está determinada
    if numero_2 != 0:
        #Cálculo de la división
```

```
56     division=numero_1/numero_2
57     #Impresión de la división
58     print(numero_1, "/", numero_2, "=", division)
59 #División para 0
60 else:
61     #División indeterminada
62     print(numero_1, "/", numero_2, "= Indeterminado")
```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:

```
Ingrese el número 1:79
Ingrese el número 2:4
Las operaciones básicas con los números ingresados son:
79.0 + 4.0 = 83.0
79.0 - 4.0 = 75.0
79.0 * 4.0 = 316.0
79.0 / 4.0 = 19.75
```

Fig. 87. Operaciones Básicas con dos números

3.25 Hallar Datos de un círculo

Análisis El Análisis a continuación:

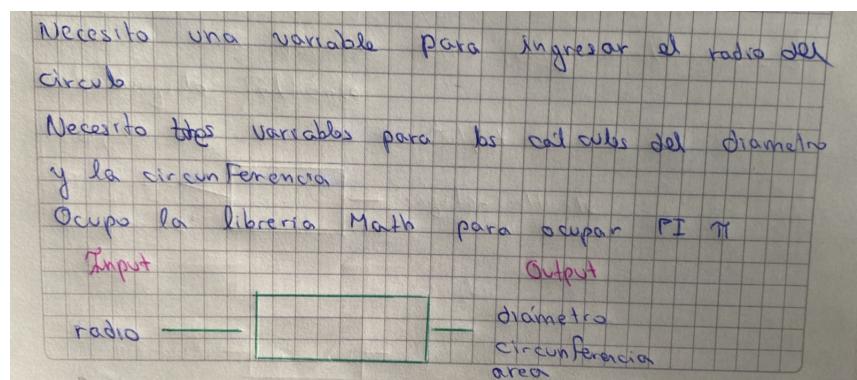


Fig. 88. Análisis Hallar Datos de un círculo

Modelo El Modelo es el siguiente:

Si y solo si radio > 0.

diametro = radio * 2.

circunferencia = 2 · radio · π

area = π · radio².

Fig. 89. Modelo Hallar Datos de un círculo

Programación Siguiendo el análisis y el modelo:

```

1 def circulo():
2     """
3         Función para encontrar el diámetro, la circunferencia y el
4         área del círculo
5     -----
6         Parámetro:
7             radio_circulo: Dato ingresado por el usuario
8
9         Retorna:
10            diámetro
11            circunferencia
12            área
13            """
14
15     #Bucle para la variable radio_circulo
16     while True:
17         #Sentencia try/except
18         try:
19             #El usuario ingresa el radio del círculo
20             radio_circulo=float(input("Ingrese el radio de un
21             círculo:"))
22             #break
23             break
24         #Si el usuario ingresa un dato inválido
25         except ValueError:
26             #Se le notifica al usuario que se equivocó

```

```

24         print("Ingrese un dato correcto!")
25 #Si el radio existe
26 if radio_circulo > 0:
27     #Cálculo del diámetro
28     diametro=radio_circulo*2
29     #Se imprime la solución del diámetro
30     print("El diámetro del círculo de radio",radio_circulo
31     , "es:",diametro,"u")
32     #Cálculo de la circunferencia
33     circunferencia=2*radio_circulo*math.pi
34     #Impresión de la circunferencia
35     print("El perímetro de la circunferencia es:",
36     circunferencia,"u")
37     #Cálculo del área del círculo
38     area=math.pi*(radio_circulo**2)
39     #Se imprime el resultado del área del círculo
40     print("El área del círculo es:",area,"u^2")
41 #Si no existe
42 else:
43     #Se le notifica al usuario que no es posible realizar
44     #los cálculos
45     print("El radio no existe, elija un valor mayor que 0"
46 )
47     #Se alienta al usuario a empezar de nuevo
48     print("Inténtelo de nuevo!")
49     #Se empieza de nuevo
50     circulo()

```

Programacion

Casos de Prueba y Ejecución del programa La ejecución es la siguiente:

```

Ingrese el radio de un círculo:450
El diámetro del círculo de radio 450.0 es: 900.0 u
El perímetro de la circunferencia es: 2827.4333882308138 u
El área del círculo es: 636172.512351933 u^2

```

Fig. 90. Encontrar Datos de un círculo

Cuando el usuario ingresa datos ilógicos, se le notifica:

```
Ingrese el radio de un círculo:-78
El radio no existe, elija un valor mayor que 0
Inténtelo de nuevo!
```

Fig. 91. Encontrar Datos de un círculo inválido

4 Discusión

Los resultados arrojados son los requeridos, a pesar de que en los cálculos no está presente la utilización de cifras significativas. La validación y verificación en el ingreso de datos es la base primordial de cada ejercicio ya que brinda al usuario una experiencia más completa, mostrándole la existencia o no de ciertos procesos al utilizar números negativos, en problemas como el cálculo de áreas, densidad, longitudes no se pueden utilizar números negativos; mientras que en problemas que involucran temperatura, cálculo de expresiones o aritmética, se puede utilizar cualquier valor numérico, aún cuando existen divisiones indeterminadas o divisiones para cero.

5 Conclusiones

Python ha ido en ascenso a lo largo de los años, y está de moda, haciendo temblar esos lenguajes que parecen inmortales. Es un lenguaje de programación fácil de aprender, perfecto para principiantes. Tiene muchos usos y un sinfín de usos, tiene una comunidad muy activa lo que garantiza que el lenguaje se mantendrá actualizado con el tiempo y aparecerán nuevas bibliotecas que nos permitirán ahorrar tiempo y trabajo. Además, como hemos visto, Python ofrece muchas ventajas a todos sus usuarios, entre ellas: es muy simple, flexible y fácil de aprender. Con una sintaxis sencilla y un buen uso de los espacios de sangría, hacen que aprender, leer e incluso compartir sea muy fácil.