Dirección Provincial de
Educación Técnico Profesional
DPETP
Secretaría Privada
Dirección General de Cultura y Educación
Provincia de Buenos Aires
Calle 13 N° 868 - La Plata
(0221) 423-5835 ; 425-7043

Instituto Superior de Formación Técnica 151

# TECNICATURA SUPERIOR

# EN ANALISIS DE SISTEMAS

## Cátedra: ANALISIS MATEMATICO II

*"Trabajo Práctico Final 2022"*

**Docente a cargo: Julio Enrique Riera**

**Alumnos:**

- **López, Walter Omar**
- **Villanueva, Beatriz Elena**

**15-DIC-2022**

# Indice

# Consigna

El siguiente trabajo práctico consiste en un programa en lenguaje Python, que permite calcular integrales definidas de tres tipos de funciones diferentes y predeterminadas, permitiendo al usuario seleccionar los coeficientes que caracterizan cada función, así como los límites de integración.

*Especificaciones solicitadas*

- De las funciones:
  - Primera función:
    - $F(x) = a \cdot x^n$
      - a Є R y n Є Z, ingresados por el usuario
      - Límites de integración variables y enteros, ingresados por el usuario.
      - Error menor a $1 \cdot 10^{-6}$
  - Segunda función:
    - $F(x) = a \cdot x^2 + b \cdot x + c$
      - a, b, c Є R, ingresados por el usuario
      - Límites de integración variables y enteros, ingresados por el usuario.
      - Error menor a $1 \cdot 10^{-6}$
  - Tercera función:
    - $F(x) = A \cdot \text{sen}\ (k \cdot x)$
      - A, k Є R, ingresados por el usuario
      - Límites de integración variables y expresados en fracciones de π, ingresados por el usuario.
      - Error menor a $1 \cdot 10^{-6}$
- Del programa:
  - Debe entregar como resultado los siguientes datos:
    - La integral de la función elegida con los límites de integración.
    - El resultado de la integración, es decir, el área obtenida.
    - El error cometido en la integración.

# Marco teórico

Python es un lenguaje de programación creado por Guido van Rossum, programador de computación de los Países Bajos, en los años 90.

Python comenzó en 1989 en el Centrum Wiskunde & Informatica (CWI), en principio como un proyecto de afición para mantenerse ocupado durante las vacaciones de Navidad. El nombre del lenguaje se inspiró en el programa de televisión de la BBC "Monty Python's Flying Circus" debido a que Guido Van Rossum era un gran aficionado del programa.

## Historial de lanzamientos de Python

- Guido Van Rossum publicó la primera versión del código Python (versión 0.9.0) en 1991. Dicha versión ya incluía buenas características, como algunos tipos de datos y funciones para la gestión de errores.
- Python 1.0 se lanzó en 1994 con nuevas funciones para procesar fácilmente una lista de datos, como la asignación, el filtrado y la reducción.
- Python 2.0 se lanzó el 16 de octubre de 2000, con nuevas características útiles para los programadores, como la compatibilidad con los caracteres Unicode y una forma más corta de recorrer una lista.
- El 3 de diciembre de 2008, se lanzó Python 3.0. Incluía características como la función de impresión y más soporte para la división de números y la gestión de errores.

# Justificación

## Beneficios

Los beneficios de Python incluyen los siguientes:

- Los desarrolladores pueden leer y comprender fácilmente los programas en Python debido a su sintaxis básica similar a la del inglés.
- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa con menos líneas de código en comparación con muchos otros lenguajes.
- Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++, entre otros.
- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrarse con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix.

## Características

Como características representativas podemos decir que es un lenguaje de alto nivel, fácil de utilizar, interpretado, con tipado dinámico, fuertemente tipado, multiplataforma y que está orientado a objetos.

- Es un lenguaje de alto nivel porque es más cercano a los idiomas humanos que otros lenguajes de programación. Por lo tanto, los programadores no deben preocuparse sobre sus funcionalidades subyacentes, como la arquitectura y la administración de la memoria.

- Es un lenguaje fácil de utilizar, ya que utiliza palabras similares a las del inglés. A diferencia de otros lenguajes de programación, Python no utiliza llaves y en su lugar, utiliza sangría.
- Un lenguaje interpretado o de script significa que ejecuta directamente el código línea por línea. Si existen errores en el código del programa, su ejecución se detiene. Así, los programadores pueden encontrar errores en el código con rapidez. Se ejecuta mediante un programa intermedio que se llama intérprete. Los lenguajes interpretados no se compilan a lenguaje de máquina para poder ejecutarse.
- El tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a ser contenido en una variable porque Python los determina en el tiempo de ejecución. Debido a esto, es posible escribir programas de Python con mayor rapidez.
- Fuertemente tipado se debe a que una vez asignado el tipo de dato a la variable este no puede ser tratado de una manera distinta al que tiene.
- Es multiplataforma porque se puede trasladar por diferentes sistemas operativos como Windows, macOS, Linux y Unix.
- Está orientado a los objetos porque considera todo como un objeto, pero también admite otros tipos de programación, como la programación estructurada y la funcional.

## Bibliotecas

Una biblioteca es una colección de códigos usados con frecuencia que los desarrolladores pueden incluir en sus programas de Python para evitar tener que escribir el código desde cero. De forma predeterminada, Python incluye la biblioteca estándar, que contiene una gran cantidad de funciones reutilizables.

Además, más de 137 000 bibliotecas de Python están disponibles para diversas aplicaciones, incluidos el desarrollo web, la ciencia de datos y el machine learning (ML).

Las más populares son:

- Matplotlib: Los desarrolladores la utilizan para trazar los datos en gráficos de dos y tres dimensiones (2D y 3D) de alta calidad. Por lo general, se utiliza en las aplicaciones científicas. Con Matplotlib, puede visualizar los datos mostrándolos en diferentes gráficos, como los gráficos de barras y los de líneas. También puede trazar varios gráficos de una sola vez, y estos se pueden trasladar a todas las plataformas.
- Pandas: Proporciona estructuras de datos optimizadas y flexibles que se pueden utilizar para manipular datos de serie temporal y datos estructurados, como las tablas y las matrices. Por ejemplo, para leer, escribir, combinar, filtrar y agrupar datos. Muchas personas lo utilizan para las tareas de ciencia de datos, análisis de datos y ML.
- NumPy: Es una conocida biblioteca que utilizan los desarrolladores para crear y administrar matrices, manipular formas lógicas y efectuar operaciones de álgebra lineal con facilidad. Admite la integración a muchos lenguajes, como C y C++.
- SymPy: Es una biblioteca de Python para matemáticas simbólicas. Su propósito es llegar a ser un sistema de álgebra por computadora (CAS) completo manteniendo el código tan simple como sea posible para poder ser legible y extensible de manera fácil. SymPy está escrito en Python enteramente. Se utiliza en este trabajo.
- Requests: Proporciona funciones útiles que se necesitan para el desarrollo web. Puede usarla para enviar solicitudes HTTP; agregar encabezados, parámetros de URL y datos; y llevar a cabo muchas más tareas cuando se comunica con aplicaciones web.
- OpenCV-Python: Se utilizan para procesar imágenes para las aplicaciones de visión artificial. Proporciona muchas funciones para las tareas de procesamiento de imágenes, como la lectura y la escritura simultáneas de imágenes, la creación de un entorno 3D a partir de uno 2D y la captura y el análisis de las imágenes de video.
- Keras: Es la biblioteca de red neuronal profunda de Python que cuenta con un excelente soporte para el procesamiento de datos, su visualización y mucho más. Admite muchas redes neuronales. Posee una estructura modular que ofrece flexibilidad en la escritura de aplicaciones innovadoras.

## Marcos

Un marco de Python es una colección de paquetes y módulos. Un módulo es un conjunto de código relacionado, y un paquete es un conjunto de módulos.

Los desarrolladores pueden usar los marcos de Python para crear aplicaciones de Python más rápido debido a que no tienen que preocuparse por los detalles de nivel inferior, como la forma en que se producen las comunicaciones en la aplicación web o el modo en que Python hará que el programa sea más rápido. Python tiene dos tipos de marcos:

- El marco de pila completa incluye casi todo lo que se necesita para crear una aplicación grande.
- El micromarco es un marco básico que proporciona funcionalidades mínimas para crear aplicaciones de Python simples. También proporciona extensiones si las aplicaciones necesitan funciones más sofisticadas.

Los desarrolladores pueden utilizar varios marcos de Python para que su desarrollo sea eficiente, incluidos los siguientes:

- Django: Es uno de los marcos web de Python de pila completa más utilizados para el desarrollo de aplicaciones web a gran escala. Proporciona varias características útiles, incluidos un servidor web para el desarrollo y las pruebas, un motor de plantillas para crear el sitio web de frontend y diversos mecanismos de seguridad.
- Flask: Es un micromarco que se utiliza para el desarrollo de aplicaciones web pequeñas. Sus características incluyen un importante soporte de la comunidad, documentación bien escrita, un motor de plantillas, pruebas de unidad y un servidor web integrado. También proporciona extensiones para el soporte de validación, las capas de asignación de bases de datos y la seguridad web. Se utiliza en este trabajo.
- TurboGears: Es un marco diseñado para crear aplicaciones web con mayor rapidez y facilidad. Estas son algunas de sus características clave:
- Estructura específica de tabla de base de datos
- Herramientas para la creación y la administración de proyectos
- Motor de plantillas para crear las bases de datos
- Motor de plantillas para crear el frontend
- Mecanismos para manejar la seguridad web
- Apache MXNet: Es un marco de aprendizaje profundo rápido, flexible y escalable que los desarrolladores utilizan para crear prototipos de investigación y aplicaciones de aprendizaje profundo. Admite múltiples lenguajes de programación, incluidos Java, C++, R y Perl. Proporciona un completo conjunto de herramientas y bibliotecas para brindar soporte al desarrollo. Por ejemplo, puede encontrar un libro interactivo de machine learning (ML), kits de herramientas de visión artificial y modelos de aprendizaje profundo para el procesamiento de lenguaje natural (NLP), que procesan este lenguaje, como el texto y el habla.
- PyTorch: Es un marco para el machine learning que se ha creado sobre la biblioteca Torch, que es otra biblioteca de machine learning de código abierto. Los desarrolladores lo utilizan para aplicaciones como las de NLP, robótica y visión artificial, para encontrar información significativa en las imágenes y los videos. También lo utilizan para ejecutar esas aplicaciones en las CPU y las GPU.

## Ejecución de la app e impresión por pantalla

Una vez se ejecuta el programa, el usuario debe abrir una página con la dirección http://localhost:5000/.

Como se muestra en la Figura 1, inicialmente el usuario se encuentra con la pantalla donde deberá seleccionar el tipo de integral que desea resolver, y luego ingresar los coeficientes necesarios para su resolución, así como los límites de integración.

Adicional a esto, incluimos que el usuario elija la cantidad de intervalos en la que se dividirá el área para obtener así el error correspondiente a la función, pudiendo ver si es suficiente hasta obtener el solicitado por la cátedra.

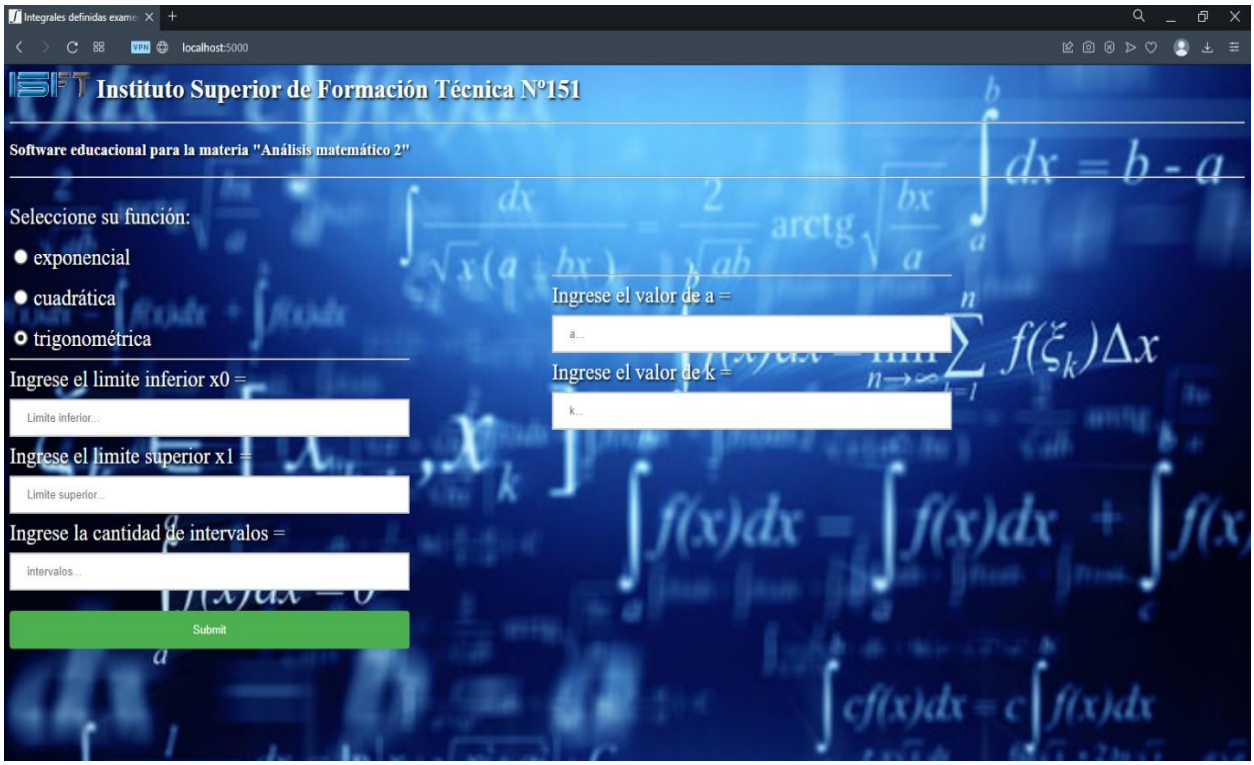



*Figura 1: Pantalla inicial*

Una vez ingresados los valores, como se ve en la Figura 2 (con un ejemplo de función trigonométrica), se envía para su resolución.

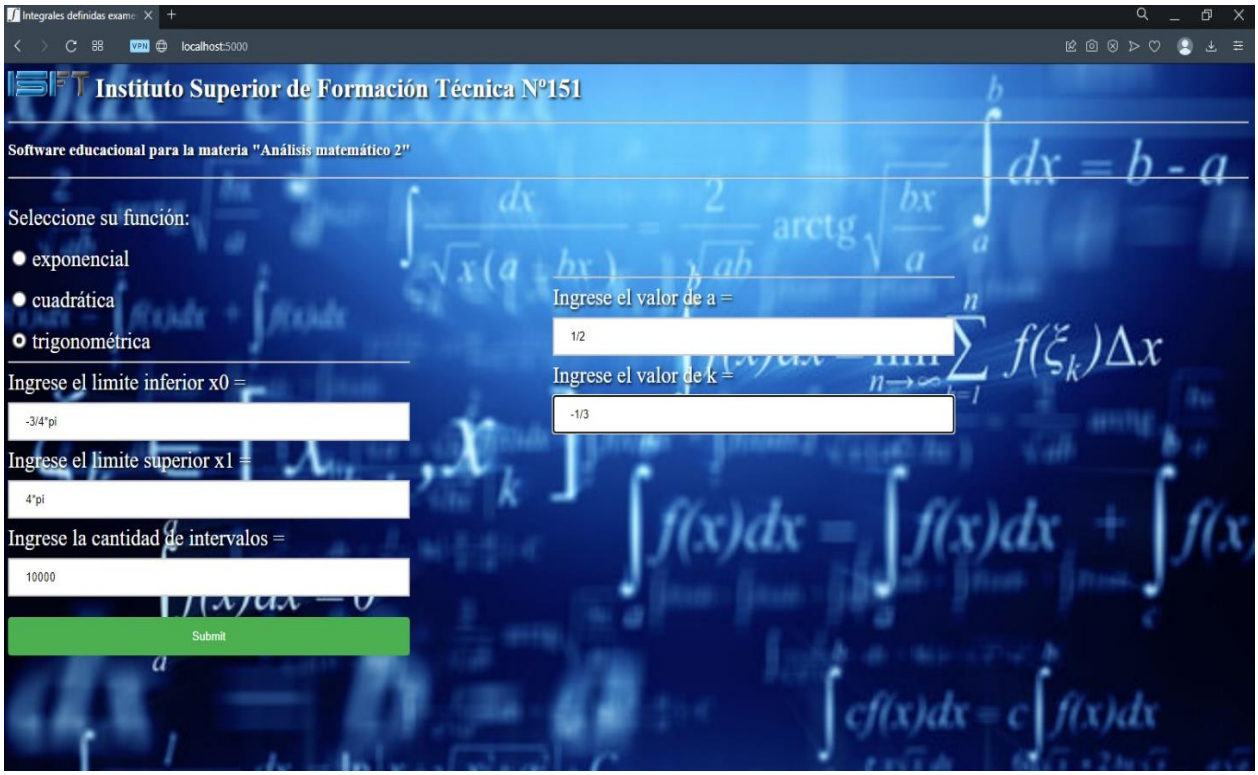



*Figura 2: Pantalla inicial con carga de coeficientes, límites e intervalos*

En la siguiente imagen (Figura 3) el programa devuelve la información completa de la integral con la función ingresada, sus límites, y la solución a la misma, como su aproximación y el error.

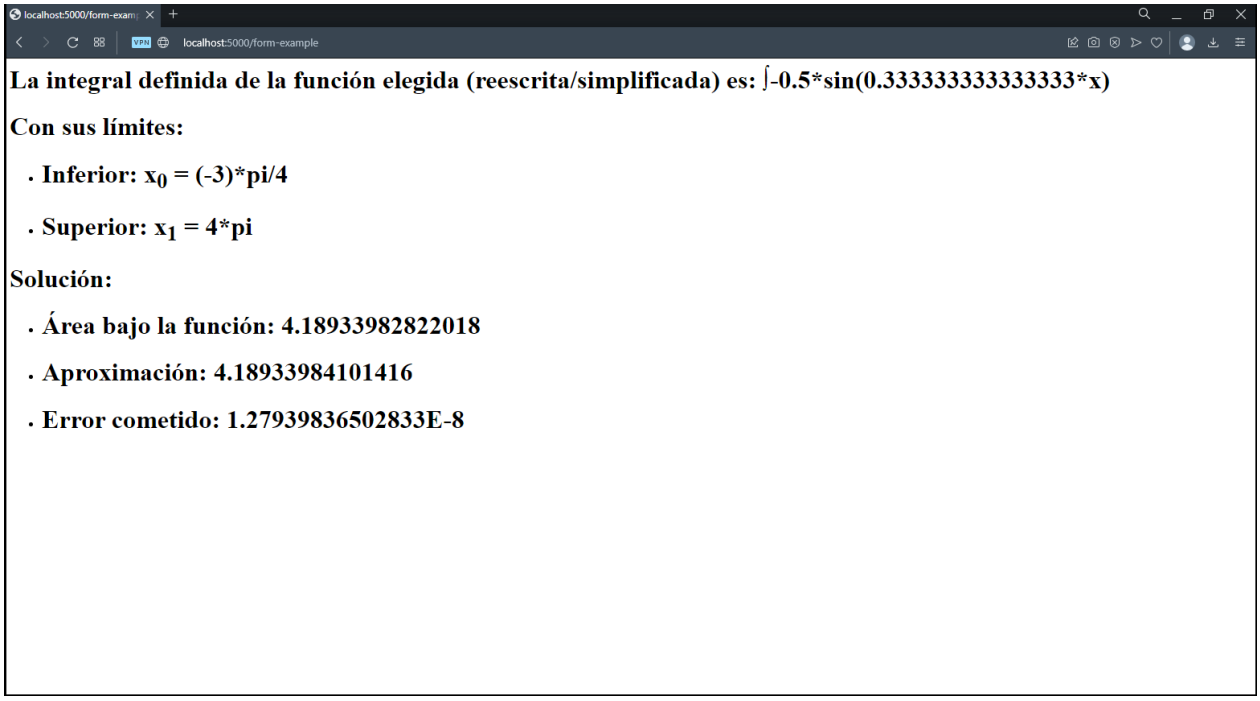



*Figura 3: Pantalla final con información completa de la función ingresada y resultados solicitados*

## Contenido de la app


TP_Final2022_Lopez_Villanueva

static

favicon.ico

Logo_ISFT_151.png

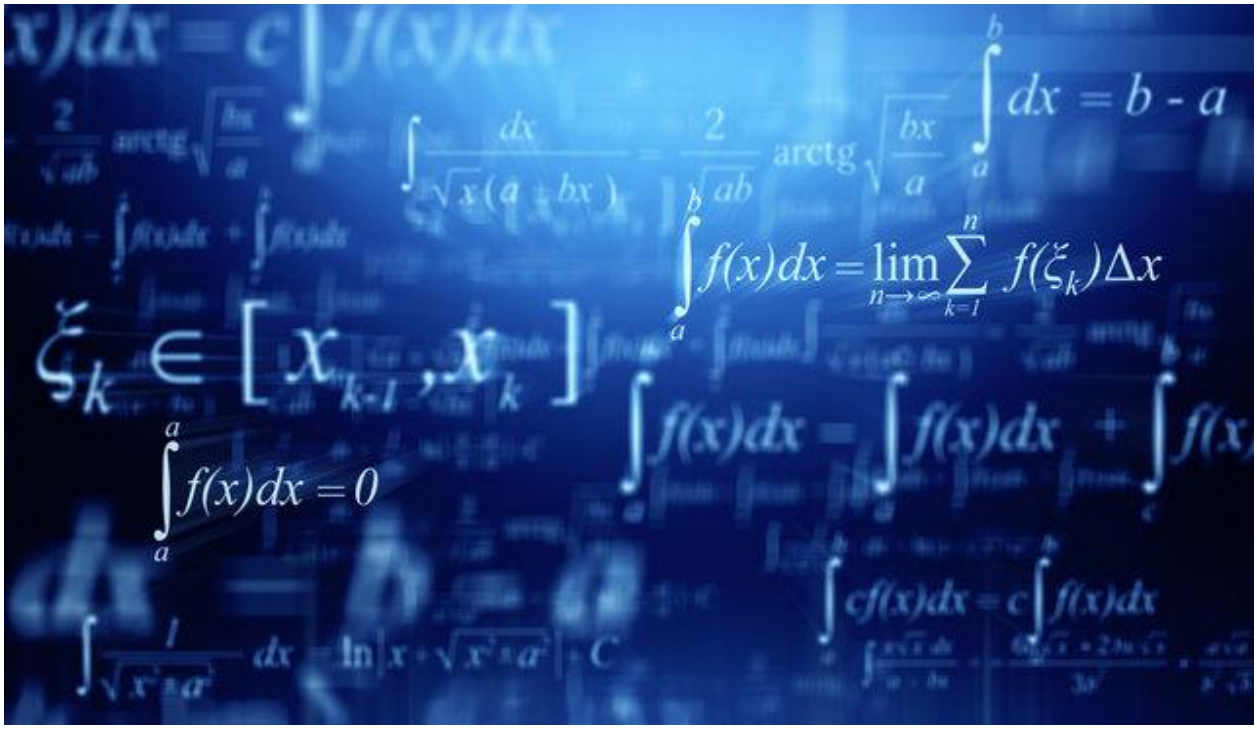
#style.css

```
h1
{
    font-size: 30px;
}
label
{
    font-size: 25px ;
}
input[type=submit] {
    width: 100%;
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-top: 15px;
    margin-bottom: 0px;
}

input[type=submit]:hover
{
    background-color: #1E90FF;
    animation: mymove 5s infinite;
}

input[type=text],select
{
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 2px solid #ccc;
    box-sizing: border-box;
}

.FirstDiv
{
    height: 500 px;
    width: 500px;
    margin-top: 30px;
}
.radio
{

    font-size: 10px;
    background-color: #1E90FF;
}
.input-radio
```

```
{
    cursor: pointer;
}

.container-animacion-1 {

    position: fixed;
    width: 150px;
    top: 0;
    right: -160px;
    bottom: 0;
    /* background: orange; */
    transition: 0.2s ease;
}

.container-animacion-2 {
    position: fixed;
    width: 150px;
    top: 0;
    right: -160px;
    bottom: 0;
    background: blue;
    transition: 0.2s ease;
}

.container-animacion-3 {
    position: fixed;
    width: 150px;
    top: 0;
    right: -160px;
    bottom: 0;
    background: green;
    transition: 0.2s ease;
}

#control-div-1:checked~.container-animacion-1
{
    position: absolute;
    top: 460px;
    left: 60%;
    width: 500px;
    transform: translate(-50%, -70%);

}

#control-div-2:checked~.container-animacion-2 {
    right: 0
}

#control-div-3:checked~.container-animacion-3 {
```

```
    right: 0
}

body
{
    background-image: url("./walpapper.jpg");
    background-size: cover;
    color: white;
    text-shadow: black 0.1em 0.1em 0.2em
}

h1.Title:before
{
    content: url(Logo_ISFT_151.png)
}
```

wallpaper.jpg




templates

index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- <link rel="icon" href="./favicon.jpg"> -->
    <link rel="shortcut icon" href={{ url_for('static',filename =
'favicon.ico')}}>
```

```
    <link rel="stylesheet" href={{ url_for('static',filename = 'style.css')}}>

    <title>Integrales definidas examen final</title>
</head>
<body>

    <h1 class="Title"> Instituto Superior de Formación Técnica Nº151</h1>
    <hr />
    <h3>Software educacional para la materia "Análisis matemático 2"</h3>
    <hr />

    <div class="FirstDiv">
        <form action="/form-example" method="POST">
            <label>Seleccione su función: </label><br><br>
            <div class="input-radio" ><input type='radio' id="control-div-1"
class="radio" style = "height: 18px; width: 18px;" name='function'
value='exponencial'> <label>exponencial</label>
                <div class="container-animacion-1">
                    <hr>
                    <div><label>Ingrese el valor de a = <input type="text"
name="aExponencial" placeholder="a..."></label></div>
                    <div><label>Ingrese el valor de n = <input type="text"
name="n" placeholder="n..."></label></div>
                </div>
            </div>
            <br>
            <div class="input-radio"><input type='radio' id="control-div-1"
class="radio" style = "height: 18px; width: 18px;" name='function'
value='cuadrática' checked> <label>cuadrática</label>

                <div class="container-animacion-1">
                   <hr>
                    <div><label>Ingrese el valor de a = <input type="text"
name="a" placeholder="a..."></label></div>
                    <div><label>Ingrese el valor de b = <input type="text"
name="b" placeholder="b..."></label></div>
                    <div><label>Ingrese el valor de c = <input type="text"
name="c" placeholder="c..."></label></div>
                </div>

            </div>
            <br>
            <div class="input-radio"><input type='radio' id="control-div-
1"  class="radio" style = "height: 18px; width: 18px;" name='function'
value='trigonométrica'> <label>trigonométrica</label>

                <div class="container-animacion-1">
                    <hr>
                    <div><label>Ingrese el valor de a = <input type="text"
name="aSeno" placeholder="a..."></label></div>
```

```
                    <div><label>Ingrese el valor de k = <input type="text"
name="k" placeholder="k..."></label></div>
                </div>

            </div>
            <hr />
            <!-- <div class="input-radio" ><input type="radio" id="control-div-1"
name="controlar-divs"><label for="control-div-1">Controlar div 1</label>
                <div class="container-animacion-1">
                    <p>div a controlar</p>
                </div>
            </div> -->

            <!-- <div><label>Ingrese la funcion f(x) =  <input type="text"
name="f" placeholder="&int;Función..."></label></div> -->
            <div><label>Ingrese el limite inferior x0 = <input type="text"
name="x0" placeholder="Limite inferior..."></label></div>
            <div><label>Ingrese el limite superior x1 = <input type="text"
name="x1" placeholder="Limite superior..."></label></div>
            <div><label>Ingrese la cantidad de intervalos = <input type="text"
name="intervalos" placeholder="intervalos..."></label></div>
            <input type="submit" value="Submit" class="Button">

        </form>
    </div>
</body>
</html>
```

app01.py

```
# import main Flask class and request object
from flask import Flask, render_template, request
from sympy import *

# create the Flask app
app = Flask(__name__)


@app.route('/')
def template():
    return render_template("index.html")


def midpoint(f, a, b, n):
    error = 1*10**-6
    h = float(b - a)/n  # finding midpoint
    result = 0
    x = a + (h / 2)
    for _ in range(n):
        result += f(x)  # adding all the f(x)
        if(result > error):
```

```
            print(error)
        x += h
    result *= h  # multiplying with the midpoint
    return result


def quadraticArea(aq, bq, c, a, b, intervalo):
    x = symbols('x')
    aquadratic = int(aq)
    bquadratic = int(bq)
    cquadratic = int(c)
    f0 = aquadratic*x**2 + bquadratic*x + cquadratic
    f = str(f0)
    raiz1, raiz2 = solve(f)
    raiz1cast = round(raiz1,13)
    raiz2cast = round(raiz2,13)
    area = []

    def f2(x): return eval(f)

    if (a < raiz1 and b <= raiz1):
        rIntegral = integrate(f, (x, a, b))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, a, b, intervalo)
        area.append(abs(aproximacion))

    if (a < raiz1 and b > raiz1 and b <= raiz2):
        r1Integral = abs(integrate(f, (x, a, raiz1cast)))
        r2Integral = abs(integrate(f, (x, raiz1cast, b)))
        valueOne = midpoint(f2, a, raiz1cast, intervalo)
        valueTwo = midpoint(f2, raiz1cast, b, intervalo)
        aproximacion = abs(valueOne - valueTwo)
        area.append(abs(r1Integral + r2Integral))
        area.append(aproximacion)

    if (a < raiz1 and b > raiz2):
        r1Integral = abs(integrate(f, (x, a, raiz1cast)))
        r2Integral = abs(integrate(f, (x, raiz1cast, raiz2cast)))
        r3Integral = abs(integrate(f, (x, raiz2cast, b)))
        valueOne = midpoint(f2, a, raiz1cast, intervalo)
        valueTwo = midpoint(f2, raiz1cast, raiz2cast, intervalo)
        valueThree = midpoint(f2, raiz2cast, b, intervalo)
        aproximacion = abs(valueOne - valueTwo+valueThree)
        area.append(r1Integral + r2Integral + r3Integral)
        area.append(aproximacion)

    if (a >= raiz1 and a < raiz2 and b > raiz1 and b <= raiz2):
        rIntegral = integrate(f, (x, a, b))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, a, b, intervalo)
```

```
        area.append(abs(aproximacion))

    if (a >= raiz1 and a < raiz2 and b > raiz2):
        r1Integral = abs(integrate(f, (x, a, raiz2cast)))
        r2Integral = abs(integrate(f, (x, raiz2cast, b)))
        area.append(abs(r1Integral + r2Integral))
        valueOne = midpoint(f2, a, raiz2cast, intervalo)
        valueTwo = midpoint(f2, raiz2cast, b, intervalo)
        aproximacion = abs(valueOne - valueTwo)
        area.append(aproximacion)

    if (raiz1 == raiz2):
        rIntegral = integrate(f, (x, a, b))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, a, b, intervalo)
        area.append(abs(aproximacion))

    area.append(abs(area[0] - area[1]))
    return area


def firstFunction(aExp, nExp, x0, b,intervalo):
    x = symbols('x')
    aExponencial = float(aExp)
    nExponencial = int(nExp)
    f0 = aExponencial*x**nExponencial
    f = str(f0)
    exponente = nExponencial
    area = []
    a = aExponencial

    def f2(x): return eval(f)

    parImpar = ''

    if (exponente % 2 == 0):
        parImpar = "es par"
    else:
        parImpar = "es impar"

    if (parImpar == 'es par' and a > 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, b)))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, x0, b, intervalo)
        area.append(abs(aproximacion))

    if (parImpar == 'es par' and a < 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, b)))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, x0, b, intervalo)
```

```
        area.append(abs(aproximacion))

    if (parImpar == 'es impar' and a < 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, 0)))
        r2Integral = abs(integrate(f, (x, 0, b)))
        area.append(abs(rIntegral - r2Integral))
        valueOne = midpoint(f2, x0, 0, intervalo)
        valueTwo = midpoint(f2, 0, b, intervalo)
        aproximacion = abs(valueOne + valueTwo)
        area.append(aproximacion)

    if (parImpar == 'es impar' and a > 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, 0)))
        r2Integral = abs(integrate(f, (x, 0, b)))
        area.append(abs(rIntegral - r2Integral))
        valueOne = midpoint(f2, x0, 0, intervalo)
        valueTwo = midpoint(f2, 0, b, intervalo)
        aproximacion = abs(valueOne + valueTwo)
        area.append(aproximacion)

    area.append(abs(area[0] - area[1]))

    return area


def sinFunction(aseno, kseno, a, b, intervalo):
    x = symbols('x')
    contadorA = 0
    contadorB = 0
    area = []
    aSenoCasteado = float(eval(aseno))
    k = abs(float(eval(kseno)))
    f = aSenoCasteado*sin(k*x)
    f0 = str(f)
    def f2(x): return eval(f0)
    acasteado = float(eval("a"))
    bcasteado = float(eval("b"))
    picasteado = float(eval("pi"))

    if (a>=0):
        while (contadorA*pi/k <= a):
            contadorA_x_Pi_K = contadorA * pi / k
            if (contadorA_x_Pi_K <= a and a < ((contadorA + 1) * pi / k)):
                if (a < b and b <= ((contadorA + 1) * pi / k)):
                    rIntegral = abs(integrate(f, (x, acasteado, bcasteado)))
                    area.append(rIntegral)
                    aproximacion = abs(midpoint(f2, acasteado, bcasteado, intervalo))
                    area.append(abs(aproximacion))
                else:
                    rIntegral = abs(integrate(f, (x, acasteado, (contadorA + 1) *
```

```
                    picasteado / k)))
                    valueOne = abs(midpoint(f2, acasteado,
                    (contadorA + 1) * picasteado / k, intervalo))
            contadorA += 1
    else:
        while(contadorA * pi / k >= a):
            contadorA_x_Pi_K = contadorA * pi / k
            if(contadorA_x_Pi_K >= a and a > ((contadorA - 1)* pi / k)):
                if(a < b and b <= ((contadorA) * pi / k)):
                    rIntegral = abs(integrate(f,(x, acasteado, bcasteado)))
                    area.append(rIntegral)
                    aproximacion = abs(midpoint(f2, acasteado, bcasteado, intervalo))
                    area.append(abs(aproximacion))
                else:
                    rIntegral = abs(integrate(f,(x, acasteado, contadorA *
                    picasteado / k)))
                    # area.append(rIntegral)
                    valueOne = abs(midpoint(f2, acasteado,
                    contadorA * picasteado / k, intervalo))
            contadorA -= 1
        contadorA = contadorA + 1
    contadorB = contadorA

    if (b > contadorB * pi / k):
        while (contadorB * pi / k < b):
            if (contadorB * pi / k < b and b <= ((contadorB + 1) * pi / k)):
                r2Integral = abs(integrate(f, (x, contadorB * picasteado / k,
                 bcasteado)))
                valueTwo = abs(midpoint(f2, contadorB * picasteado / k, bcasteado,
                 intervalo))
            contadorB += 1
        contadorB2 = contadorB - 1
        cant_semiperiodos = contadorB2 - contadorA
        r3Integral = (abs(integrate(f, (x, 0, picasteado / k)))
        * cant_semiperiodos)
        area.append(rIntegral + r2Integral + r3Integral)
        valueThree = abs(midpoint(f2, 0, picasteado / k, intervalo)
        * cant_semiperiodos)
        aproximacion = abs(valueOne + valueTwo + valueThree)
        area.append(aproximacion)

    area.append(abs(area[0] - area[1]))

    return area


@app.route('/form-example', methods=['GET', 'POST'])
def form_example():

    if request.method == 'POST':
```

```
        init_printing()
        x = symbols('x')

        f = request.form.get('f')
        x0 = request.form.get('x0')
        x1 = request.form.get('x1')
        Intervals = request.form.get('intervalos')

        ax = request.form.get('a')
        bx = request.form.get('b')
        c = request.form.get('c')

        aExp = request.form.get('aExponencial')
        nExp = request.form.get('n')

        aseno = request.form.get('aSeno')
        k = request.form.get('k')
        typeFunction = request.form.get('function')

        if (typeFunction == 'cuadrática'):
            x = symbols('x')
            intervalsCasteado = int(Intervals)
            x0Casteado = int(x0)
            x1Casteado = int(x1)
            ListIntegralAproximation = quadraticArea(ax, bx, c, x0Casteado,
            x1Casteado, intervalsCasteado)
            integralDefinida = ListIntegralAproximation[0]
            aproximation = ListIntegralAproximation[1]
            axCasteado = int(ax)
            bxCasteado = int(bx)
            cCasteado = int(c)
            f = axCasteado*x**2 + bxCasteado*x + cCasteado
            integral = integrate(f, x)
            notationScientific = ListIntegralAproximation[2]
            # f2 = lambda x: eval(f)
            # aproximation = midpoint(f2,x0Casteado,x1Casteado,intervalo)
        elif (typeFunction == 'exponencial'):
            x = symbols('x')
            intervalsCasteado = int(Intervals)
            evalx0 = eval(x0)
            evalx1 = eval(x1)
            x0Casteado = float(evalx0)
            x1Casteado = float(evalx1)
            ListIntegralAproximation = firstFunction(aExp, nExp, x0Casteado,
            x1Casteado, intervalsCasteado)
            integralDefinida = ListIntegralAproximation[0]
            aproximation = ListIntegralAproximation[1]
            aExpCasteado = int(aExp)
            nExpCasteado = int(nExp)
            f = aExpCasteado*x**nExpCasteado
```

```
            # f = str(f3)
            # f2 = lambda x: eval(f)
            # aproximation = midpoint(f2,x0Casteado,x1Casteado,10000)
            integral = integrate(f, x)
            notationScientific = ListIntegralAproximation[2]
        elif (typeFunction == 'trigonométrica'):
            x = symbols('x')
            intervalsCasteado = int(Intervals)
            evalx0 = eval(x0)
            evalx1 = eval(x1)
            x0Casteado = float(evalx0)
            x1Casteado = float(evalx1)

            # def f2(x): return eval(f)
            # aproximation = midpoint(f2, x0Casteado, x1Casteado, 10000)
            aSenoCasteado = float(eval(aseno))
            KCasteado = float(eval(k))
            f = aSenoCasteado * sin(KCasteado * x)
            ListIntegralAproximation = sinFunction(aseno, k, evalx0, evalx1,
            intervalsCasteado)
            integralDefinida = ListIntegralAproximation[0]
            aproximation = ListIntegralAproximation[1]
            integral = integrate(f, x)
            notationScientific = ListIntegralAproximation[2]

        return '''
                <h1>La integral definida de la función elegida (reescrita/simplificada)
                es: &int;{}</h1>
                <h1>Con sus límites:</h1>
                <ul>
                    <li><h1>Inferior: x<sub>0</sub> = {}</h1></li>
                    <li><h1>Superior: x<sub>1</sub> = {}</h1></li>
                </ul>
                <h1>Solución:</h1>
                <ul>
                    <li><h1>Área bajo la función: {}</h1></li>
                    <li><h1>Aproximación: {}</h1></li>
                    <li><h1>Error cometido: {}</h1></li>
                    </ul>
                '''.format(f, x0, x1,integralDefinida, aproximation,
                notationScientific)


if __name__ == '__main__':
    # run app in debug mode on port 5000
    app.run(debug=True, port=5000)

# import main Flask class and request object
from flask import Flask, render_template, request
from sympy import *
```

```
# create the Flask app
app = Flask(__name__)


@app.route('/')
def template():
    return render_template("index.html")


def midpoint(f, a, b, n):
    error = 1*10**-6
    h = float(b - a)/n  # finding midpoint
    result = 0
    x = a + (h / 2)
    for _ in range(n):
        result += f(x)  # adding all the f(x)
        if(result > error):
            print(error)
        x += h
    result *= h  # multiplying with the midpoint
    return result


def quadraticArea(aq, bq, c, a, b, intervalo):
    x = symbols('x')
    aquadratic = int(aq)
    bquadratic = int(bq)
    cquadratic = int(c)
    f0 = aquadratic*x**2 + bquadratic*x + cquadratic
    f = str(f0)
    raiz1, raiz2 = solve(f)
    raiz1cast = round(raiz1,13)
    raiz2cast = round(raiz2,13)
    area = []
    def f2(x): return eval(f)
    if (a < raiz1 and b <= raiz1):
        rIntegral = integrate(f, (x, a, b))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, a, b, intervalo)
        area.append(abs(aproximacion))
    if (a < raiz1 and b > raiz1 and b <= raiz2):
        r1Integral = abs(integrate(f, (x, a, raiz1cast)))
        r2Integral = abs(integrate(f, (x, raiz1cast, b)))
        valueOne = midpoint(f2, a, raiz1cast, intervalo)
        valueTwo = midpoint(f2, raiz1cast, b, intervalo)
        aproximacion = abs(valueOne - valueTwo)
        area.append(abs(r1Integral + r2Integral))
        area.append(aproximacion)
    if (a < raiz1 and b > raiz2):
        r1Integral = abs(integrate(f, (x, a, raiz1cast)))
```

```
        r2Integral = abs(integrate(f, (x, raiz1cast, raiz2cast)))
        r3Integral = abs(integrate(f, (x, raiz2cast, b)))
        valueOne = midpoint(f2, a, raiz1cast, intervalo)
        valueTwo = midpoint(f2, raiz1cast, raiz2cast, intervalo)
        valueThree = midpoint(f2, raiz2cast, b, intervalo)
        aproximacion = abs(valueOne - valueTwo+valueThree)
        area.append(r1Integral + r2Integral + r3Integral)
        area.append(aproximacion)

    if (a >= raiz1 and a < raiz2 and b > raiz1 and b <= raiz2):
        rIntegral = integrate(f, (x, a, b))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, a, b, intervalo)
        area.append(abs(aproximacion))
    if (a >= raiz1 and a < raiz2 and b > raiz2):
        r1Integral = abs(integrate(f, (x, a, raiz2cast)))
        r2Integral = abs(integrate(f, (x, raiz2cast, b)))
        area.append(abs(r1Integral + r2Integral))
        valueOne = midpoint(f2, a, raiz2cast, intervalo)
        valueTwo = midpoint(f2, raiz2cast, b, intervalo)
        aproximacion = abs(valueOne - valueTwo)
        area.append(aproximacion)
    if (raiz1 == raiz2):
        rIntegral = integrate(f, (x, a, b))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, a, b, intervalo)
        area.append(abs(aproximacion))

    area.append(abs(area[0] - area[1]))
    return area


def firstFunction(aExp, nExp, x0, b,intervalo):
    x = symbols('x')
    aExponencial = float(aExp)
    nExponencial = int(nExp)
    f0 = aExponencial*x**nExponencial
    f = str(f0)
    exponente = nExponencial
    area = []
    a = aExponencial

    def f2(x): return eval(f)

    parImpar = ''

    if (exponente % 2 == 0):
        parImpar = "es par"
    else:
        parImpar = "es impar"
```

```
    if (parImpar == 'es par' and a > 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, b)))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, x0, b, intervalo)
        area.append(abs(aproximacion))
    if (parImpar == 'es par' and a < 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, b)))
        area.append(abs(rIntegral))
        aproximacion = midpoint(f2, x0, b, intervalo)
        area.append(abs(aproximacion))
    if (parImpar == 'es impar' and a < 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, 0)))
        r2Integral = abs(integrate(f, (x, 0, b)))
        area.append(abs(rIntegral - r2Integral))
        valueOne = midpoint(f2, x0, 0, intervalo)
        valueTwo = midpoint(f2, 0, b, intervalo)
        aproximacion = abs(valueOne + valueTwo)
        area.append(aproximacion)
    if (parImpar == 'es impar' and a > 0 and exponente > 0):
        rIntegral = abs(integrate(f, (x, x0, 0)))
        r2Integral = abs(integrate(f, (x, 0, b)))
        area.append(abs(rIntegral - r2Integral))
        valueOne = midpoint(f2, x0, 0, intervalo)
        valueTwo = midpoint(f2, 0, b, intervalo)
        aproximacion = abs(valueOne + valueTwo)
        area.append(aproximacion)

    area.append(abs(area[0] - area[1]))

    return area


def sinFunction(aseno, kseno, a, b, intervalo):
    x = symbols('x')
    contadorA = 0
    contadorB = 0
    area = []
    aSenoCasteado = float(eval(aseno))
    k = abs(float(eval(kseno)))
    f = aSenoCasteado*sin(k*x)
    f0 = str(f)
    def f2(x): return eval(f0)
    acasteado = float(eval("a"))
    bcasteado = float(eval("b"))
    picasteado = float(eval("pi"))

    if (a>=0):
        while (contadorA*pi/k <= a):
            contadorA_x_Pi_K = contadorA * pi / k
```

```
        if (contadorA_x_Pi_K <= a and a < ((contadorA + 1) * pi / k)):
            if (a < b and b <= ((contadorA + 1) * pi / k)):
                rIntegral = abs(integrate(f, (x, acasteado, bcasteado)))
                area.append(rIntegral)
                aproximacion = abs(midpoint(f2, acasteado, bcasteado, intervalo))
                area.append(abs(aproximacion))
            else:
                rIntegral = abs(integrate(f, (x, acasteado, (contadorA + 1) *
                picasteado / k)))
                valueOne = abs(midpoint(f2, acasteado,
                (contadorA + 1) * picasteado / k, intervalo))

        contadorA += 1
else:
    while(contadorA * pi / k >= a):
        contadorA_x_Pi_K = contadorA * pi / k
        if(contadorA_x_Pi_K >= a and a > ((contadorA - 1)* pi / k)):
            if(a < b and b <= ((contadorA) * pi / k)):
                rIntegral = abs(integrate(f,(x, acasteado, bcasteado)))
                area.append(rIntegral)
                aproximacion = abs(midpoint(f2, acasteado, bcasteado, intervalo))
                area.append(abs(aproximacion))
            else:
                rIntegral = abs(integrate(f,(x, acasteado, contadorA *
                picasteado / k)))
                # area.append(rIntegral)
                valueOne = abs(midpoint(f2, acasteado,
                contadorA * picasteado / k, intervalo))
        contadorA -= 1
    contadorA = contadorA + 1

contadorB = contadorA

if (b > contadorB * pi / k):
    while (contadorB * pi / k < b):
        if (contadorB * pi / k < b and b <= ((contadorB + 1) * pi / k)):
            r2Integral = abs(integrate(f, (x, contadorB * picasteado / k,
             bcasteado)))
            valueTwo = abs(midpoint(f2, contadorB * picasteado / k, bcasteado,
             intervalo))

        contadorB += 1
    contadorB2 = contadorB - 1
    cant_semiperiodos = contadorB2 - contadorA
    r3Integral = (abs(integrate(f, (x, 0, picasteado / k)))
    * cant_semiperiodos)
    area.append(rIntegral + r2Integral + r3Integral)
    valueThree = abs(midpoint(f2, 0, picasteado / k, intervalo)
    * cant_semiperiodos)
    aproximacion = abs(valueOne + valueTwo + valueThree)
```

```
        area.append(aproximacion)

    area.append(abs(area[0] - area[1]))

    return area


@app.route('/form-example', methods=['GET', 'POST'])
def form_example():

    if request.method == 'POST':
        init_printing()
        x = symbols('x')

        f = request.form.get('f')
        x0 = request.form.get('x0')
        x1 = request.form.get('x1')
        Intervals = request.form.get('intervalos')

        ax = request.form.get('a')
        bx = request.form.get('b')
        c = request.form.get('c')

        aExp = request.form.get('aExponencial')
        nExp = request.form.get('n')

        aseno = request.form.get('aSeno')
        k = request.form.get('k')
        typeFunction = request.form.get('function')

        if (typeFunction == 'cuadrática'):
            x = symbols('x')
            intervalsCasteado = int(Intervals)
            x0Casteado = int(x0)
            x1Casteado = int(x1)
            ListIntegralAproximation = quadraticArea(ax, bx, c, x0Casteado,
            x1Casteado, intervalsCasteado)
            integralDefinida = ListIntegralAproximation[0]
            aproximation = ListIntegralAproximation[1]
            axCasteado = int(ax)
            bxCasteado = int(bx)
            cCasteado = int(c)
            f = axCasteado*x**2 + bxCasteado*x + cCasteado
            integral = integrate(f, x)
            notationScientific = ListIntegralAproximation[2]
            # f2 = lambda x: eval(f)
            # aproximation = midpoint(f2,x0Casteado,x1Casteado,intervalo)
        elif (typeFunction == 'exponencial'):
            x = symbols('x')
            intervalsCasteado = int(Intervals)
```

```
            evalx0 = eval(x0)
            evalx1 = eval(x1)
            x0Casteado = float(evalx0)
            x1Casteado = float(evalx1)
            ListIntegralAproximation = firstFunction(aExp, nExp, x0Casteado,
            x1Casteado, intervalsCasteado)
            integralDefinida = ListIntegralAproximation[0]
            aproximation = ListIntegralAproximation[1]
            aExpCasteado = int(aExp)
            nExpCasteado = int(nExp)
            f = aExpCasteado*x**nExpCasteado
            # f = str(f3)
            # f2 = lambda x: eval(f)
            # aproximation = midpoint(f2,x0Casteado,x1Casteado,10000)
            integral = integrate(f, x)
            notationScientific = ListIntegralAproximation[2]
        elif (typeFunction == 'trigonométrica'):
            x = symbols('x')
            intervalsCasteado = int(Intervals)
            evalx0 = eval(x0)
            evalx1 = eval(x1)
            x0Casteado = float(evalx0)
            x1Casteado = float(evalx1)

            # def f2(x): return eval(f)
            # aproximation = midpoint(f2, x0Casteado, x1Casteado, 10000)
            aSenoCasteado = float(eval(aseno))
            KCasteado = float(eval(k))
            f = aSenoCasteado * sin(KCasteado * x)
            ListIntegralAproximation = sinFunction(aseno, k, evalx0, evalx1,
            intervalsCasteado)
            integralDefinida = ListIntegralAproximation[0]
            aproximation = ListIntegralAproximation[1]
            integral = integrate(f, x)
            notationScientific = ListIntegralAproximation[2]

        return '''
                <h1>La integral definida de la función elegida (reescrita/simplificada)
                es: &int;{}</h1>
                <h1>Con sus límites:</h1>
                <ul>
                    <li><h1>Inferior: x<sub>0</sub> = {}</h1></li>
                    <li><h1>Superior: x<sub>1</sub> = {}</h1></li>
                </ul>
                <h1>Solución:</h1>
                <ul>
                    <li><h1>Área bajo la función: {}</h1></li>
                    <li><h1>Aproximación: {}</h1></li>
                    <li><h1>Error cometido: {}</h1></li>
                    </ul>
```

```
                '''.format(f, x0, x1,integralDefinida, aproximation,
                notationScientific)


if __name__ == '__main__':
    # run app in debug mode on port 5000
    app.run(debug=True, port=5000)
```

# Bibliografía

https://www.sympy.org/es/

https://flask.palletsprojects.com/en/2.2.x/

https://www.python.org/

https://aws.amazon.com/es/what-is/python/

https://github.com/Omarlopezw/Definite_Integral_Solver

https://github.com/mariavuvu/Program-to-solve-definite-integrals.git