Documentacion de cuaderno de practicas

Versión 1.0

Enrique Gómez

08 de enero de 2025

Contenidos:

CAPÍTULO 1

Practica 3

1.1 Practica 3.1.1

Calcular el area de un circulo de radio r

pr3.1_1.**area_circulo**(*r*)

Calcula el área de un círculo dado su radio.

Parámetros
r (*float*) – El radio del círculo.

Devuelve
El área del círculo calculada como pi * r^2.

Tipo del valor devuelto
float

pr3.1_1.**main**()

Solicita al usuario el radio de un círculo, calcula el área y muestra el resultado en pantalla.

1.2 Practica 3.1.2

pr3.1_2.**main**()

Solicita al usuario el radio de una esfera, calcula su volumen y muestra el resultado en pantalla.

pr3.1_2.**volumen_esfera**(*r*)

Calcula el volumen de una esfera dado su radio.

Parámetros
r (*float*) – El radio de la esfera.

Devuelve
El volumen de la esfera calculado como (4/3) * pi * r^3.

Tipo del valor devuelto
float

1.3 Practica 3.1.3

pr3.1_3.**main**()

Solicita al usuario los coeficientes de una ecuación de primer grado y muestra su solución.

pr3.1_3.**solucion_ecuacion_primero_grado**(*a*, *b*)

Calcula la solución de una ecuación de primer grado de la forma ax + b = 0.

Parámetros

- **a** (*float*) – El coeficiente de la variable x.
- **b** (*float*) – El término independiente.

Devuelve

Una cadena que describe la solución:

- Si *a != 0*, retorna el valor de x.
- Si *a == 0* y *b == 0*, indica que la ecuación tiene infinitas soluciones.
- Si *a == 0* y *b != 0*, indica que no hay solución.

Tipo del valor devuelto
str

1.4 Practica 3.1.4

pr3.1_4.**main**()

Solicita al usuario los coeficientes de una ecuación de segundo grado y muestra las soluciones de la ecuación.

pr3.1_4.**resolver_ecuacion_segundo_grado**(*a*, *b*, *c*)

Resuelve una ecuación de segundo grado de la forma ax^2 + bx + c = 0.

Parámetros

- **a** (*float*) – El coeficiente de x^2.
- **b** (*float*) – El coeficiente de x.
- **c** (*float*) – El término independiente.

Devuelve

Mensaje indicando las soluciones de la ecuación:

- Si el discriminante es positivo, retorna las dos soluciones reales.
- Si el discriminante es cero, retorna una solución real.
- Si el discriminante es negativo, indica que no tiene soluciones reales.

Tipo del valor devuelto
str

1.5 Practica 3.1.5

pr3.1_5.**calcular_porcentaje**(*a*, *b*)

Calcula el porcentaje de una parte con respecto a un total.

Parámetros

- **a** (*float*) – El valor de la parte.
- **b** (*float*) – El valor del total.

Devuelve

El porcentaje calculado de la parte con respecto al total. str: Un mensaje indicando si el cálculo fue posible o no.

Tipo del valor devuelto

float

pr3.1_5.**main**()

Solicita al usuario los valores de la parte y el total, y calcula el porcentaje. Si el total es cero, pide al usuario que ingrese un valor válido.

1.6 Practica 3.2.1

pr3.2_1.**analizar_lista_numeros**(*numeros*)

Analiza una lista de números y devuelve el número mayor y menor de la lista.

Parámetros

numeros (*list*) – Una lista de números enteros.

Devuelve

El número mayor y menor de la lista.

Tipo del valor devuelto

tuple

pr3.2_1.**celsius_a_kelvin**(*celsius*)

Convierte grados Celsius a Kelvin.

La fórmula para la conversión es: K = °C + 273

Parámetros

celsius (*float*) – La temperatura en grados Celsius.

Devuelve

La temperatura equivalente en grados Kelvin.

Tipo del valor devuelto

float

pr3.2_1.**es_primo**(*num*)

Determina si un número es primo.

Un número primo es aquel que solo es divisible por 1 y por sí mismo.

Parámetros

num (*int*) – El número a verificar.

Devuelve
True si el número es primo, False si no lo es.

Tipo del valor devuelto
bool

pr3.2_1.**main**()

Función principal que solicita al usuario una serie de entradas y realiza las tareas: 1. Verifica si un número es primo. 2. Convierte una temperatura de grados Celsius a Kelvin. 3. Encuentra el número mayor y menor en una lista de números.

1.7 Practica 3.2.2

pr3.2_2.**convertir_numero**()

Convierte un número entero a cadena y viceversa, mostrando los tipos de datos en cada paso.

Solicita al usuario un número entero, lo convierte a cadena, lo muestra con su tipo, y luego lo convierte nuevamente a entero y muestra el tipo de nuevo.

pr3.2_2.**convertir_y_calcular_cuadrado**()

Convierte un número decimal representado como cadena a tipo flotante, luego calcula y muestra el cuadrado de ese número.

Solicita al usuario un número decimal como cadena, lo convierte a flotante y calcula su cuadrado.

pr3.2_2.**eliminar_duplicados**()

Convierte una lista de palabras a un conjunto para eliminar duplicados, y luego vuelve a convertir el conjunto en una lista.

Solicita al usuario una lista de palabras separadas por espacios, elimina los duplicados convirtiéndola en un conjunto, y luego vuelve a convertir el conjunto a una lista.

pr3.2_2.**main**()

Función principal que ejecuta las tres tareas de conversión de tipos de datos: 1. Convierte un número entero a cadena y viceversa. 2. Convierte un número decimal en cadena a flotante y calcula su cuadrado. 3. Elimina duplicados de una lista de palabras usando un conjunto y luego la convierte de nuevo a lista.

1.8 Practica 3.3.1

pr3.3_1.**calcular_cuadrado**()

Solicita al usuario un número real, calcula su cuadrado y verifica si el resultado es un número entero.

El programa calcula el cuadrado de un número real, luego verifica si el resultado es entero y muestra un mensaje indicando si es entero o no.

pr3.3_1.**es_entero**(*numero*)

Verifica si un número es entero.

Parámetros
numero (*float*) – El número que se va a verificar.

Devuelve
True si el número es entero, False si no lo es.

Tipo del valor devuelto
bool

1.9 Practica 3.3.2

pr3.3_2.**verificar_condiciones_logicas**()

Realiza dos verificaciones lógicas utilizando operadores “and” y “or”.

Primero, evalúa si ambas condiciones son verdaderas usando el operador “and”: - Si “a” es mayor que 2 y “b” es menor que 15.

Luego, evalúa si al menos una de las condiciones es verdadera usando el operador “or”: - Si el usuario es premium o tiene un cupón.

Los resultados de ambas verificaciones se imprimen por separado.

CAPÍTULO 2

Practica 4

2.1 Practica 4.1

pr4.1.**calcular_perimetro_area_cuadrado**(*lado*)

Calcula el perímetro y el área de un cuadrado dado el valor de su lado.

Parámetros
lado (*float*) – El valor del lado del cuadrado.

Devuelve

Un tuple que contiene dos valores:

- El perímetro del cuadrado (float).
- El área del cuadrado (float).

Tipo del valor devuelto
tuple

2.2 Practica 4.2

pr4.2.**calcular_perimetro_area_rectangulo**(*lado1*, *lado2*)

Calcula el perímetro y el área de un rectángulo dado el valor de sus dos lados.

Parámetros

- **lado1** (*float*) – El valor del primer lado del rectángulo.
- **lado2** (*float*) – El valor del segundo lado del rectángulo.

Devuelve

Un tuple que contiene dos valores:

- El perímetro del rectángulo (float).

- El área del rectángulo (float).

Tipo del valor devuelto
tuple

2.3 Practica 4.3

pr4.3.**calcular_area_triangulo**(*base*, *altura*)

Calcula el área de un triángulo dado su base y altura.

Parámetros

- **base** (*float*) – El valor de la base del triángulo.
- **altura** (*float*) – El valor de la altura del triángulo.

Devuelve
El área del triángulo calculada con la fórmula (base * altura) / 2.

Tipo del valor devuelto
float

2.4 Practica 4.4

pr4.4.**calcular_area_perimetro_triangulo**(*lado1*, *lado2*, *lado3*)

Calcula el área y el perímetro de un triángulo dado sus tres lados utilizando la fórmula de Herón para el área.

Parámetros

- **lado1** (*float*) – El valor del primer lado del triángulo.
- **lado2** (*float*) – El valor del segundo lado del triángulo.
- **lado3** (*float*) – El valor del tercer lado del triángulo.

Devuelve

Un tuple que contiene dos valores:

- El perímetro del triángulo (float).
- El área del triángulo (float).

Tipo del valor devuelto
tuple

2.5 Practica 4.5

pr4.5.**registrar_persona**()

Solicita los datos de una persona, como nombre, apellidos, edad y estatura, y luego muestra la información ingresada.

La función no toma parámetros ni devuelve valores. Solicita la entrada del usuario a través de la consola y muestra los resultados en formato legible.

Ejemplo de ejecución:
Introduce tu nombre: John Introduce tus apellidos: Doe Introduce tu edad: 30 Introduce tu estatura (en metros): 1.75

Salida:
Información de la persona: Nombre: John Apellidos: Doe Edad: 30 años Estatura: 1.75 metros

2.6 Practica 4.6

pr4.6.**calcular_area_trapecio**(*base_mayor*, *base_menor*, *altura*)

Calcula el área de un trapecio dado su base mayor, base menor y altura.

La fórmula utilizada es:
Área = ((base mayor + base menor) * altura) / 2

Parámetros

- **base_mayor** (*float*) – La longitud de la base mayor del trapecio.
- **base_menor** (*float*) – La longitud de la base menor del trapecio.
- **altura** (*float*) – La altura del trapecio.

Devuelve
El área del trapecio.

Tipo del valor devuelto
float

2.7 Practica 4.7

pr4.7.**intercambiar_valores**(*a*, *b*)

Intercambia los valores de dos variables.

La función toma dos números, intercambia sus valores y los devuelve.

Parámetros

- **a** (*float*) – El primer número.
- **b** (*float*) – El segundo número.

Devuelve
Una tupla con los valores de “b” y “a” intercambiados.

Tipo del valor devuelto
tuple

2.8 Practica 4.8

pr4.8.**desglosar_segundos**(*segundos_totales*)

Desglosa una cantidad de segundos en su equivalente en horas, minutos y segundos.

La función toma una cantidad total de segundos y devuelve su equivalente en horas, minutos y segundos utilizando división entera y el operador módulo.

Parámetros
segundos_totales (*int*) – La cantidad de segundos a desglosar.

Devuelve
Una tupla con las horas, minutos y segundos correspondientes.

Tipo del valor devuelto
tuple

2.9 Practica 4.9

pr4.9.**desglosar_segundos**(*segundos_totales*)

Desglosa una cantidad de segundos en su equivalente en horas, minutos y segundos.

La función toma una cantidad total de segundos y devuelve su equivalente en horas, minutos y segundos utilizando división entera y el operador módulo.

Parámetros
segundos_totales (*int*) – La cantidad de segundos a desglosar.

Devuelve
Una tupla con las horas, minutos y segundos correspondientes.

Tipo del valor devuelto
tuple

2.10 Practica 4.10

pr4.10.**convertir_a_pesetas**(*euros*)

Convierte una cantidad en euros a pesetas.

Esta función toma una cantidad en euros y la convierte a pesetas utilizando el tipo de cambio fijo de 1 euro = 166386 pesetas.

Parámetros
euros (*float*) – La cantidad en euros a convertir.

Devuelve
La cantidad equivalente en pesetas.

Tipo del valor devuelto
float

CAPÍTULO 3

Practica 5

3.1 Practica 5.1

pr5.1.**dividir**(*dividendo*, *divisor*)

Realiza la división de dos números, verificando que el divisor no sea cero.

Esta función recibe dos números: el dividendo y el divisor, y devuelve el resultado de la división. Si el divisor es cero, la función devuelve un mensaje indicando que no es posible realizar la división.

Parámetros

- **dividendo** (*float*) – El número que será dividido.
- **divisor** (*float*) – El número que divide al dividendo.

Devuelve

El resultado de la división si el divisor es diferente de cero,
o un mensaje de error si el divisor es cero.

Tipo del valor devuelto
float or str

3.2 Practica 5.2

pr5.2.**evaluar_beneficios**(*ingresos: int*, *gastos: int*) → int

Calcula los beneficios de una empresa a partir de sus ingresos y gastos.

Esta función evalúa si los beneficios de una empresa son positivos, negativos o nulos, y devuelve un valor entero dependiendo de la situación de la empresa.

- Si los beneficios son positivos, imprime «La empresa es solvente» y devuelve +1.
- Si los beneficios son cero, imprime «Se ha alcanzado el punto de equilibrio» y devuelve 0.

- Si los beneficios son negativos, imprime «La empresa esta en numeros rojos» y devuelve -1.

Parámetros

- **ingresos** (*int*) – Los ingresos de la empresa.
- **gastos** (*int*) – Los gastos de la empresa.

Devuelve
+1 si la empresa es solvente (beneficios positivos). 0 si se ha alcanzado el punto de equilibrio (beneficios cero). -1 si la empresa está en números rojos (beneficios negativos).

Tipo del valor devuelto
int

3.3 Practica 5.3

pr5.3.**deteccion**(*a: int*, *b: int*, *c: int*) → bool

Detecta si tres números se han introducido en orden creciente.

La función evalúa si los tres números dados están en orden creciente, es decir, si el primero es menor que el segundo y el segundo es menor que el tercero.

Parámetros

- **a** (*int*) – Primer número.
- **b** (*int*) – Segundo número.
- **c** (*int*) – Tercer número.

Devuelve
True si los números están en orden creciente. False si no están en orden creciente.

Tipo del valor devuelto
bool

3.4 Practica 5.4

pr5.4.**dia_de_la_semana**(*dia: int*) → str

Dada una entrada numérica entre 1 y 7, devuelve el correspondiente día de la semana.

La función recibe un número entero entre 1 y 7 e imprime el nombre del día de la semana correspondiente.

Parámetros
dia (*int*) – Un número entero entre 1 y 7, donde: 1 = lunes, 2 = martes, ..., 7 = domingo.

Devuelve
El nombre del día de la semana correspondiente al número proporcionado.

Tipo del valor devuelto
str

Muestra
ValueError – Si el número ingresado no está entre 1 y 7.

3.5 Practica 5.5

pr5.5.**menu_principal**()

Muestra un menú interactivo al usuario con diferentes opciones. Según la opción seleccionada, se muestra un mensaje indicando la acción seleccionada o un mensaje de error si la opción no es válida.

El menú tiene las siguientes opciones: 1. Cargar fichero de datos 2. Almacenar fichero de datos 3. Modificar datos 4. Salir

El programa seguirá mostrando el menú hasta que el usuario elija la opción "Salir".

3.6 Practica 5.6

pr5.6.**tabla_multiplicar**(*numero*)

Muestra la tabla de multiplicar de un número natural dado.

Parámetros: numero (int): El número natural cuya tabla de multiplicar se va a mostrar.

Si el número dado es negativo, se muestra un mensaje indicando que debe ser un número natural. De lo contrario, muestra la tabla de multiplicar del número proporcionado, desde 1 hasta 10.

3.7 Practica 5.7

pr5.7.**sumar_numeros_entre**(*inicio*, *fin*, *metodo*)

Calcula la suma de los números enteros entre dos números dados, utilizando un bucle "for" o "while".

Parámetros: inicio (int): El número inicial. fin (int): El número final. metodo (str): El método a utilizar para la suma ("for" o "while").

Retorna: int: La suma de los números entre "inicio" y "fin" inclusive, según el método seleccionado.

3.8 Practica 5.8

pr5.8.**calcular_factorial**(*n*)

Calcula el factorial de un número entero no negativo.

Parámetros: n (int): Número entero no negativo para calcular su factorial.

Retorna: int: El factorial de *n*.

Levanta: ValueError: Si *n* es un número negativo.

3.9 Practica 5.9

pr5.9.**estadisticas_numeros**()

Solicita al usuario una serie de números y calcula las estadísticas más relevantes: el número más grande, el más pequeño y la media.

El proceso termina cuando el usuario ingresa 0. El número 0 no se incluye en los cálculos.

La función muestra por pantalla: - El número más grande - El número más pequeño - La media de los números ingresados

3.10 Practica 5.10

pr5.10.**es_numero_perfecto**(*n*)

Determina si un número entero dado es perfecto.

Un número perfecto es aquel que es igual a la suma de sus divisores propios (excluyendo el mismo número). Por ejemplo, el número 6 es perfecto porque sus divisores son 1, 2 y 3, y 1 + 2 + 3 = 6.

Parámetros: n (int): El número a verificar.

Retorna: bool: True si el número es perfecto, False si no lo es.

3.11 Practica 5.11

pr5.11.**calcular_producto_digitos**(*numero: int*) → int

Calcula el producto de los dígitos de un número entero.

Dado un número entero, la función calcula el producto de todos sus dígitos. Por ejemplo, con el número 123, el producto sería 1 * 2 * 3 = 6.

Parámetros: numero (int): El número entero cuyo producto de dígitos se va a calcular.

Retorna: int: El producto de los dígitos del número.

pr5.11.**obtener_numero**() → int

Solicita un número entero al usuario una vez.

Retorna: int: El número entero ingresado por el usuario.

CAPÍTULO 4

Practica 6

4.1 Practica 6.1.1

pr6.1_1.**concatenar_vectores**(*N*, *M*)

Dada dos vectores de enteros, de tamaños N y M respectivamente, esta función concatena ambos vectores en un nuevo vector que tiene un tamaño de N+M.

Argumentos: N – Primer vector de enteros (array de numpy o lista de Python). M – Segundo vector de enteros (array de numpy o lista de Python).

Retorna: Un nuevo vector que es la concatenación de N y M.

4.2 Practica 6.1.2

pr6.1_2.**visualizar_hasta_valor**(*vector*, *A*)

Dado un vector de enteros de tamaño N, esta función visualiza los valores de los elementos hasta encontrar el primer elemento cuyo valor sea mayor o igual a un número A inclusive. El resto de los elementos no se visualizarán.

Argumentos: vector – El vector de enteros (array de numpy). A – El número límite para visualizar los elementos.

4.3 Practica 6.1.3

pr6.1_3.**productoEscalar**(*vectorA*, *vectorB*)

Calcula el producto escalar de dos vectores de 3 componentes.

Argumentos: vectorA – Primer vector (numpy array). vectorB – Segundo vector (numpy array).

Retorna: El resultado del producto escalar entre vectorA y vectorB.

pr6.1_3.**resta**(*vectorA*, *vectorB*)

Resta dos vectores de 3 componentes.

Argumentos: vectorA – Primer vector (numpy array). vectorB – Segundo vector (numpy array).

Retorna: El vector resultado de la resta de vectorA y vectorB.

pr6.1_3.**suma**(*vectorA*, *vectorB*)

Suma dos vectores de 3 componentes.

Argumentos: vectorA – Primer vector (numpy array). vectorB – Segundo vector (numpy array).

Retorna: El vector resultado de la suma de vectorA y vectorB.

4.4 Practica 6.1.4

pr6.1_4.**ejecutar_calculos**()

Solicita al usuario un vector de números reales y calcula el valor máximo, mínimo y la media del vector.

El usuario ingresa un número entero N, que representa la cantidad de elementos del vector, seguido de los valores reales que conforman dicho vector. Luego se calculan y muestran: - El valor máximo del vector. - El valor mínimo del vector. - La media de los valores del vector.

No retorna ningún valor, solo imprime los resultados en consola.

pr6.1_4.**maximo**(*vector*)

Devuelve el valor máximo de un vector de números reales.

Argumento: vector – Un array de números reales.

Retorna: El valor máximo del vector.

pr6.1_4.**media**(*vector*)

Devuelve la media de los valores de un vector de números reales.

Argumento: vector – Un array de números reales.

Retorna: La media de los valores del vector.

pr6.1_4.**minimo**(*vector*)

Devuelve el valor mínimo de un vector de números reales.

Argumento: vector – Un array de números reales.

Retorna: El valor mínimo del vector.

4.5 Practica 6.1.5

pr6.1_5.**llenar_y_sumar_matriz**()

Función que llena una matriz de 3x3 con valores aleatorios entre 0 y 20, imprime la matriz, y luego imprime la suma de las filas y las columnas.

4.6 Practica 6.1.6

pr6.1_6.**operaciones_matrices**()

Función que realiza operaciones con dos matrices de 3x5: - Suma - Diferencia - Producto elemento a elemento

Luego imprime los resultados de estas operaciones.

4.7 Practica 6.1.7

pr6.1_7.**resolver_sistema_ecuaciones**()

Resuelve un sistema de tres ecuaciones lineales con tres incógnitas utilizando matrices.

El sistema de ecuaciones se representa como: A * X = B donde:

> A es la matriz de coeficientes 3x3, X es el vector de incógnitas 3x1, B es el vector de términos independientes 3x1.

El programa realiza los siguientes pasos: 1. Solicita al usuario ingresar los valores de la matriz de coeficientes (A) y del vector de términos independientes (B). 2. Calcula el determinante de la matriz de coeficientes A. 3. Si el determinante de A es cero, informa que el sistema no tiene solución única. 4. Si el determinante no es cero, calcula la matriz inversa de A. 5. Resuelve el sistema de ecuaciones utilizando la fórmula X = A**-1 [1] * B. 6. Muestra la solución del sistema, es decir, el vector de incógnitas X.

Ejemplo de uso: El usuario debe ingresar las entradas de la matriz A y el vector B de la siguiente manera: - Una matriz A de 3x3 con los coeficientes de las ecuaciones. - Un vector B de 3x1 con los términos independientes.

El programa luego calcula y muestra el determinante de A, la matriz inversa de A (si es posible), y la solución del sistema de ecuaciones.

Requisitos: - El sistema debe tener una solución única, lo que implica que el determinante de la matriz A debe ser diferente de cero.

Retorna: None: La función imprime la solución en consola.

4.8 Practica 6.2.1

pr6.2_1.**contar_vocales_consonantes**(*palabra*)

Esta función toma una palabra y cuenta el número de vocales y consonantes que contiene.

Parámetros: palabra (str): La palabra sobre la cual se va a contar las vocales y consonantes.

Retorna: tuple: Un tuple con dos valores:

- El número de vocales en la palabra.
- El número de consonantes en la palabra.

Ejemplo: >>> contar_vocales_consonantes(«Hola») (2, 2)

4.9 Practica 6.2.2

pr6.2_2.**quitar_espacios**(*cadena_car*)

Esta función recibe una cadena de caracteres y elimina los espacios en blanco al principio y al final.

Parámetros: cadena_car (str): La cadena de caracteres a la cual se le eliminarán los espacios en blanco.

Retorna: str: La cadena corregida sin espacios en blanco al principio y al final.

Ejemplo: >>> quitar_espacios(» hola como estas «) “hola como estas”

4.10 Practica 6.2.3

pr6.2_3.**eliminar_a**(*cadena*)

Esta función elimina todas las ocurrencias del caracter “a” en una cadena de caracteres.

Parámetros: cadena (str): La cadena de caracteres de la cual se eliminarán todas las “a”.

Retorna: str: La cadena sin las ocurrencias del caracter “a”.

Ejemplo: >>> eliminar_a(«banana») “bn”

4.11 Practica 6.2.4

pr6.2_4.**es_palindromo**(*cadena*)

Esta función comprueba si una cadena es un palíndromo. Una palabra es un palíndromo si se lee igual de izquierda a derecha que de derecha a izquierda.

Parámetros: cadena (str): La cadena que se desea comprobar.

Retorna: bool: Retorna True si la cadena es un palíndromo, False en caso contrario.

Ejemplo: >>> es_palindromo(«reconocer») True >>> es_palindromo(«hola») False

4.12 Practica 6.3.1

pr6.3_1.**lista_metodos**()

Esta función demuestra la utilidad de varios métodos de listas en Python.

1. Ordena una lista de menor a mayor utilizando el método sort().
2. Ordena la misma lista de mayor a menor utilizando sort(reverse=True).
3. Elimina el último elemento de la lista utilizando el método pop().
4. Muestra la posición de un elemento en una lista utilizando index().
5. Añade un nuevo elemento a la lista utilizando append().
6. Elimina un elemento específico de la lista utilizando remove().

Ejemplo: >>> lista_metodos()

CAPÍTULO 5

Practica 7

5.1 Practica 7.1

pr7.1.**grafico_1**()
: Grafica la función y = 3x - x^3 en el intervalo [-10, 10].

pr7.1.**grafico_10**()
: Grafica la función y = (x - 1) * exp(-x) en el intervalo [-30, 30].

pr7.1.**grafico_11**()
: Grafica la función y = log(x) / x en el intervalo [1, 30].

pr7.1.**grafico_12**()
: Grafica la función y(x) = sin(4pix) * e^(-5x) en el intervalo [0, 1].

pr7.1.**grafico_13**()
: Grafica la función y(x) = cos(2pix) * e^(-x) en el intervalo [0, 5].

pr7.1.**grafico_2**()
: Grafica la función y = x^4 - 2x^2 - 8 en el intervalo [-10, 10].

pr7.1.**grafico_3**()
: Grafica la función y = (x^3) / (x - 1)^2 en el intervalo [-20, 20].

pr7.1.**grafico_4**()
: Grafica la función y = (x^4 + 1) / x^2 en el intervalo [-20, 20].

pr7.1.**grafico_5**()
: Grafica la función y = (x^2) / (2 - x) en el intervalo [-20, 20].

pr7.1.**grafico_6**()
: Grafica la función y = x / (1 + x^2) en el intervalo [-20, 20].

pr7.1.**grafico_7**()

Grafica la función y = (x^2 - 3x + 2) / (x^2 + 1) en el intervalo [-20, 20].

pr7.1.**grafico_8**()

Grafica la función y = x + sqrt(x) en el intervalo [-30, 30].

pr7.1.**grafico_9**()

Grafica la función y = exp(1/x) en el intervalo [-20, 20], sin incluir el valor 0.

5.2 Practica 7.2

pr7.2.**cuadrados_y_cubos**(*tupla*)

Devuelve una tupla con los elementos elevados al cuadrado y al cubo.

Parámetros: tupla (tuple): Tupla de números enteros.

Retorna: tuple: Una tupla con dos sub-tuplas: una con los cuadrados y otra con los cubos de los elementos.

pr7.2.**ejecutar_funcion_elegida**(*opcion*, *tupla*)

Ejecuta la función elegida por el usuario, proporcionando resultados basados en la opción seleccionada.

Parámetros: opcion (int): Número que representa la función que el usuario quiere ejecutar. tupla (tuple): Tupla de números que se utilizará como parámetro para la función seleccionada.

Retorna: None

pr7.2.**invertir_y_longitud**(*tupla*)

Devuelve una tupla invertida y su longitud.

Parámetros: tupla (tuple): Tupla de números.

Retorna: tuple: Una tupla con la versión invertida y la longitud de la tupla original.

pr7.2.**min_y_max**(*tupla*)

Devuelve el mínimo y el máximo de los elementos de la tupla.

Parámetros: tupla (tuple): Tupla de números enteros.

Retorna: tuple: Una tupla con el mínimo y el máximo de los elementos.

pr7.2.**pares_e_impares**(*tupla*)

Devuelve una tupla con los elementos pares e impares separados.

Parámetros: tupla (tuple): Tupla de números enteros.

Retorna: tuple: Una tupla con dos sub-tuplas: una con los números pares y otra con los números impares.

pr7.2.**suma_y_producto**(*tupla*)

Devuelve la suma y el producto de los elementos de la tupla.

Parámetros: tupla (tuple): Tupla de números enteros.

Retorna: tuple: Una tupla con la suma y el producto de los elementos.

5.3 Practica 7.3

Un diccionario en Python es una estructura de datos que almacena pares de clave-valor. Es mutable y permite un acceso rápido a los valores mediante las claves. Se define utilizando llaves {} y los pares de clave-valor se separan por comas.

Métodos útiles de los diccionarios:

1. get(): Devuelve el valor de una clave, o un valor por defecto si la clave no existe.
2. keys(): Devuelve una vista de las claves del diccionario.
3. values(): Devuelve una vista de los valores del diccionario.
4. items(): Devuelve una vista de los pares clave-valor del diccionario.
5. update(): Actualiza el diccionario con los pares clave-valor de otro diccionario.
6. pop(): Elimina una clave y devuelve su valor.

CAPÍTULO 6

Practica 8

6.1 Practica 8.1

class pr8.1.**Calculadora**(*numero1*, *numero2*)

Bases: object

La clase Calculadora permite realizar operaciones básicas entre dos números: suma, resta, multiplicación y división.

Atributos:

numero1 (float): El primer número para las operaciones. numero2 (float): El segundo número para las operaciones.

Métodos:

__init__(self, numero1, numero2): Inicializa los números para las operaciones. sumar(self): Devuelve la suma de los dos números. restar(self): Devuelve la resta entre los dos números. multiplicar(self): Devuelve la multiplicación de los dos números. dividir(self): Devuelve la división de los dos números, o un error si el segundo número es cero.

dividir()

Devuelve la división de los dos números.

Si el segundo número es 0, devuelve un mensaje de error.

multiplicar()

Devuelve la multiplicación de los dos números.

restar()

Devuelve la resta entre los dos números.

sumar()

Devuelve la suma de los dos números.

6.2 Practica 8.2

class pr8.2.**DataPlotter**(*x*, *y*)

Bases: object

La clase DataPlotter permite crear diferentes tipos de gráficos (línea, dispersión, barras) utilizando datos proporcionados.

Atributos:

x (list): Lista de valores para el eje X. y (list): Lista de valores para el eje Y. line_color (str): Color de la línea en los gráficos (por defecto "blue"). line_width (float): Grosor de la línea (por defecto 2). line_style (str): Estilo de la línea (por defecto "-"). marker (str): Tipo de marcador (por defecto None).

bar(*title='Gráfico de Barras'*, *xlabel='Eje X'*, *ylabel='Eje Y'*, *grid=True*)

Crea un gráfico de barras con las propiedades configuradas.

Parámetros

- **title** – Título de la gráfica (str)
- **xlabel** – Etiqueta del eje X (str)
- **ylabel** – Etiqueta del eje Y (str)
- **grid** – Muestra una cuadrícula si es True (bool)

plot(*title='Gráfica'*, *xlabel='Eje X'*, *ylabel='Eje Y'*, *grid=True*)

Crea una gráfica con las propiedades configuradas.

Parámetros

- **title** – Título de la gráfica (str)
- **xlabel** – Etiqueta del eje X (str)
- **ylabel** – Etiqueta del eje Y (str)
- **grid** – Muestra una cuadrícula si es True (bool)

scatter(*title='Diagrama de Dispersión'*, *xlabel='Eje X'*, *ylabel='Eje Y'*, *grid=True*)

Crea un diagrama de dispersión con las propiedades configuradas.

Parámetros

- **title** – Título de la gráfica (str)
- **xlabel** – Etiqueta del eje X (str)
- **ylabel** – Etiqueta del eje Y (str)
- **grid** – Muestra una cuadrícula si es True (bool)

set_properties(*color='blue'*, *width=2*, *style='-'*, *marker=None*)

Configura las propiedades de la gráfica.

Parámetros

- **color** – Color de la línea (str)
- **width** – Grosor de la línea (float)
- **style** – Estilo de la línea (str)
- **marker** – Tipo de marcador (str)

6.3 Practica 8.3

class pr8.3.**Cilindro**(*radio*, *altura*)

Bases: *Objeto3D*

Representa un cilindro, un objeto tridimensional.

Atributos:
radio (float): Radio de la base del cilindro. altura (float): Altura del cilindro.

area_superficie()

Calcula el área de superficie del cilindro.

Devuelve
El área de superficie del cilindro.

volumen()

Calcula el volumen del cilindro.

Devuelve
El volumen del cilindro.

class pr8.3.**Circulo**(*radio*)

Bases: *Objeto2D*

Representa un círculo, un objeto bidimensional.

Atributos:
radio (float): Radio del círculo.

area()

Calcula el área del círculo.

Devuelve
El área del círculo.

perimetro()

Calcula el perímetro del círculo.

Devuelve
El perímetro del círculo.

class pr8.3.**Cubo**(*lado*)

Bases: *Objeto3D*

Representa un cubo, un objeto tridimensional.

Atributos:
lado (float): Longitud de un lado del cubo.

area_superficie()

Calcula el área de superficie del cubo.

Devuelve
El área de superficie del cubo.

volumen()

Calcula el volumen del cubo.

Devuelve
El volumen del cubo.

class pr8.3.**Esfera**(*radio*)

Bases: *Objeto3D*

Representa una esfera, un objeto tridimensional.

Atributos:
radio (float): Radio de la esfera.

area_superficie()

Calcula el área de superficie de la esfera.

Devuelve
El área de superficie de la esfera.

volumen()

Calcula el volumen de la esfera.

Devuelve
El volumen de la esfera.

class pr8.3.**Objeto2D**

Bases: *ObjetoGeometrico*

Clase base para objetos geométricos bidimensionales (2D). Hereda de la clase ObjetoGeometrico y debe implementar los métodos para calcular el perímetro y el área en las subclases.

area()

Calcula el área del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

perimetro()

Calcula el perímetro del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

class pr8.3.**Objeto3D**

Bases: *ObjetoGeometrico*

Clase base para objetos geométricos tridimensionales (3D). Hereda de la clase ObjetoGeometrico y debe implementar los métodos para calcular el volumen y el área de superficie en las subclases.

area_superficie()

Calcula el área de superficie del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

volumen()

Calcula el volumen del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

class pr8.3.**ObjetoGeometrico**

Bases: `object`

Clase base para todos los objetos geométricos. Se usa como clase base para los objetos 2D y 3D.

class pr8.3.**Rectangulo**(*base*, *altura*)

Bases: *Objeto2D*

Representa un rectángulo, un objeto bidimensional.

Atributos:
base (float): Base del rectángulo. altura (float): Altura del rectángulo.

area()

Calcula el área del rectángulo.

Devuelve
El área del rectángulo.

perimetro()

Calcula el perímetro del rectángulo.

Devuelve
El perímetro del rectángulo.

class pr8.3.**Triangulo**(*lado1*, *lado2*, *lado3*, *altura_base*, *base*)

Bases: *Objeto2D*

Representa un triángulo, un objeto bidimensional.

Atributos:
lado1 (float): Longitud del primer lado. lado2 (float): Longitud del segundo lado. lado3 (float): Longitud del tercer lado. altura_base (float): Altura correspondiente a la base. base (float): Longitud de la base.

area()

Calcula el área del triángulo.

Devuelve
El área del triángulo.

perimetro()

Calcula el perímetro del triángulo.

Devuelve
El perímetro del triángulo.

6.4 Practica 8.4

class pr8.4.**Automovil**(*marca*, *modelo*, *consumo_combustible*, *capacidad_personas*)

Bases: *Vehiculo*

Representa un automóvil, un tipo de vehículo.

Atributos:
capacidad_personas (int): Número de personas que el automóvil puede transportar.

caracteristicas()

Devuelve las características del automóvil como un string, incluyendo la capacidad de personas.

Devuelve
Características del automóvil.

class pr8.4.**Camion**(*marca*, *modelo*, *consumo_combustible*, *capacidad_carga*)

Bases: *Vehiculo*

Representa un camión, un tipo de vehículo con capacidad de carga.

Atributos:
capacidad_carga (float): Capacidad de carga en toneladas.

caracteristicas()

Devuelve las características del camión como un string, incluyendo la capacidad de carga.

Devuelve
Características del camión.

puede_transportar(*carga*)

Determina si el camión puede transportar una carga dada.

Parámetros
carga – Carga que se desea transportar en toneladas.

Devuelve
True si el camión puede transportar la carga, False en caso contrario.

class pr8.4.**Motocicleta**(*marca*, *modelo*, *consumo_combustible*, *tipo*)

Bases: *Vehiculo*

Representa una motocicleta, un tipo de vehículo.

Atributos:
tipo (str): Tipo de motocicleta (por ejemplo, «Deportiva», «Scooter»).

caracteristicas()

Devuelve las características de la motocicleta como un string, incluyendo el tipo de motocicleta.

Devuelve
Características de la motocicleta.

class pr8.4.**Objeto2D**

Bases: *ObjetoGeometrico*

Clase base para objetos geométricos bidimensionales (2D). Hereda de la clase ObjetoGeometrico y debe implementar los métodos para calcular el perímetro y el área en las subclases.

area()

Calcula el área del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

perimetro()

Calcula el perímetro del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

class pr8.4.**Objeto3D**

Bases: *ObjetoGeometrico*

Clase base para objetos geométricos tridimensionales (3D). Hereda de la clase ObjetoGeometrico y debe implementar los métodos para calcular el volumen y el área de superficie en las subclases.

area_superficie()

Calcula el área de superficie del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

volumen()

Calcula el volumen del objeto. Este método debe ser implementado por la subclase.

Muestra
NotImplementedError – Si no está implementado en la subclase.

class pr8.4.**ObjetoGeometrico**

Bases: object

Clase base para todos los objetos geométricos. Se utiliza como clase base para los objetos 2D y 3D.

class pr8.4.**Vehiculo**(*marca*, *modelo*, *consumo_combustible*)

Bases: object

Clase base para vehículos. Proporciona métodos para calcular el costo de viaje y obtener las características.

Atributos:
marca (str): Marca del vehículo. modelo (str): Modelo del vehículo. consumo_combustible (float): Consumo de combustible en litros por kilómetro.

caracteristicas()

Devuelve las características del vehículo como un string.

Devuelve
Características del vehículo.

costo_viaje(*distancia*, *precio_combustible*)

Calcula el costo de un viaje en función de la distancia y el precio del combustible.

Parámetros

- **distancia** – Distancia a recorrer en kilómetros.
- **precio_combustible** – Precio del combustible por litro.

Devuelve
El costo del viaje.

CAPÍTULO 7

Indices y tablas:

- genindex
- modindex
- search

Índice de Módulos Python

p

Índice