

## Casos de Estudio en el TP05 (clase anterior)

Se toma EMC en clase de teoría de la semana del TP06

### Ejercicios

1. ¿Puede explicar con sus propias palabras por qué ambos programas imprimen lo mismo?

| Argumentos posicionales   | Argumentos de palabras clave  |
|---|---|
| <pre>def nombre_persona(nombre, apellido):     print(f'{nombre} {apellido}')  nombre_persona('Lionel', 'Messi')</pre> | <pre>def nombre_persona(nombre, apellido):     print(f'{nombre} {apellido}')  nombre_persona(apellido = 'Messi', nombre = 'Lionel')</pre> |

2. Analizar, ejecutar, realizar la prueba de escritorio en forma manual y con el debugger del VSC.

|   |   |
|---|---|
| Pruebe para s = 1   | Pruebe para v = 10, luego elimine el # de la última línea, debe mostrar el tiempo. ¿Cómo evita utilizar global v?   |
| <pre>def dos(s):     s = s + 5     print('Dos:', s)  def uno(s):     s = s + 10     dos(s)     print('Uno:', s)     return s  #principal s = int(input('Inicial:'))#1 s = s + uno(s) print('Global:',s)</pre> | <pre>def acelerar():     global v     tiempo = 1     v+= 5     return  # programa principal v = float(input('Velocidad?:')) print(f'Velocidad: {v} km/h') print('Aumento la velocidad!') acelerar() print(f'Velocidad: {v} km/h') #print('Tiempo:', tiempo)</pre> |

3. Analice y realice la prueba de escritorio utilizando el *debugger* de VSC

|  |  |
|--|--|
| <pre>def mi_funcion(x, y=50):     "Parámetros con valores iniciales"     print('x:', x)     print('y:', y)  a = int(input('a:')) #10 mi_funcion(a) print(mi_funcion.__doc__)</pre> | <pre>def f1():     "Juan Galan: poeta jujeño 1913-1963"      s = "Jujuy le han puesto de nombre,     debe ser cosa de Dios;     en el idioma del cielo     así se llama el amor...     "      def f2():         print(s)      f2()  f1() print(f1.__doc__)</pre> |
|--|--|

4. Analizar y ejecutar el algoritmo que calcula la serie de Taylor de la función seno de más abajo, dónde x es el valor de un ángulo (expresado en radianes) y n es el número de términos. Mostrar el esquema de los módulos

según la programación modular, y los ámbitos de las variables. También hacer la prueba de escritorio para un ángulo de 45 grados sexagesimales y 4 términos, utilice el *debugger* del VSC.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}, \forall x$$

```
def bienvenida(n):
    nro = 20
    car = '-'
    print(car*nro)
    print('Hola ', n)
    print(car*nro)
    print('Este algoritmo calcula el seno ')
    print('de un ángulo en grado sexagesimal')
    print('con la serie de Taylor hasta un ')
    print('término determinado por Ud. ')
    print(car*nro)

def verificaP():
    while True:
        x = float(input('Ángulo en sexagesimales x>=0:'))
        if x >= 0:
            return x

def verificaMenor10():
    while True:
        m10 = int(input('N° de términos [1,10]:'))
        if m10 >= 1 and m10 <= 10:
            return m10
```

```
def factorial(x):
    p = 1
    for i in range(1, x+1):
        p *= i
    return p

def potencia(base, exponente):
    p = 1
    for i in range(exponente):
        p *= base
    return p

#Principal
nombre = input('Ingrese su nombre:')
bienvenida(nombre)
a = verificaP()
x = a*pi/180 #radianes
m = verificaMenor10() #valor final términos
suma = 0
for n in range(m):
    t = 2*n+1
    suma += potencia(-1,n)/factorial(t)*potencia(x,t)
print('Seno calculado = ', suma)
print('Seno función interna = ', sin(x))
```

5. Realice dos algoritmos para calcular la siguiente fórmula:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

- Realice el algoritmo para una cantidad n de términos de la suma.
- Realice el algoritmo hasta que el término o error o aproximación sea menor a 0,001

6. Resolver:

Escribir un módulo que utiliza tres parámetros que representan un ancho y altura, el tercer parámetro es un carácter a utilizar en el dibujo de un rectángulo. En el ejemplo siguiente se leen los valores 5 (ancho), 3 (altura) con el carácter “o”, resultando el gráfico:

|   |   |   |   |   |
|---|---|---|---|---|
| o | o | o | o | o |
| o | o | o | o | o |
| o | o | o | o | o |

7. Escribir un módulo denominado PMS que tiene dos parámetros formales base y exponente. Calcular la base elevada al exponente, siendo la base un número real cualquiera y exponente un valor entero positivo o nulo. Utilizar las multiplicaciones sucesivas de la base. Si el cálculo no puede realizarse debe devolver cero (0).

8. Mediante un menú de opciones resolver:

- Ingresar dos valores a y b validando que estén en el intervalo [0,9] y mostrarlos en letras separados por una serie de asteriscos
- Si a y b son pares intercambiar los valores y mostrarlos en letras

9. Realizar un programa modular que muestre un menú de opciones. Debe ingresar inicialmente dos valores que representan el numerador y denominador de una fracción.
- sumar\_fracciones(n1,d1,n2,d2) calcula la suma de dos fracciones. El resultado es otra fracción cuyo numerador es  $n1*d2+d1*n2$  y denominador  $d1*d2$ . Se debe simplificar la fracción resultado.
  - restar\_fracciones(n1,d1,n2,d2) calcula la resta de dos fracciones. El resultado es otra fracción numerador= $n1*d2-d1*n2$  y denominador= $d1*d2$ . Se debe simplificar la fracción resultado.
  - multiplicar\_fracciones(n1,d1,n2,d2) calcula el producto de dos fracciones. El resultado es una fracción con numerador  $n1*n2$  y denominador  $d1*d2$ . Se debe simplificar la fracción resultado.
  - dividir\_fracciones(n1,d1,n2,d2) calcula la división de dos fracciones. El resultado es una fracción con numerador  $n1*d2$  y denominador  $d1*n2$ . Se debe simplificar la fracción resultado.

En cada ejecución del programa deberá ingresar la fracción solicitando numerador y denominador. La fracción debe simplificarse y mostrarse.

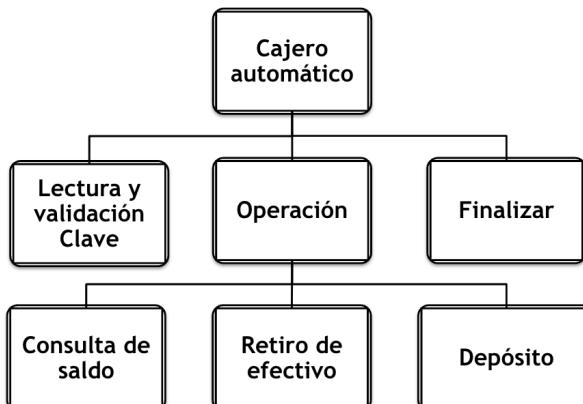
Al mostrar la fracción se debe mostrar la misma simplificada. En caso de denominador 1, sólo debe mostrar el numerador.

Puede implementar una función `simplificar_funcion(n,d)` que devuelve 2 nuevos valores resultados de la simplificación, para ello hay que dividir n y d por el mcd de ambos.

Puede implementar una función `mcd(n,d)` que devuelve el máximo común divisor de ambos parámetros.

10. Diseñar un algoritmo que simule una calculadora básica debe utilizar un módulo denominado `calcular(op,a,b)` donde los parámetros a y b son variables de tipo float y representan los operandos de la expresión aritmética y op es un parámetro de tipo cadena de caracteres que puede ser: 'suma', 'resta', 'multiplicación' y 'división', y se realiza la operación correspondiente. Mediante un menú de opciones el operador debe ingresar los datos de a y b y luego debe poder seleccionar una operación aritmética.

11. Escribir un algoritmo que simule las operaciones que se realizan en un cajero automático de acuerdo al esquema jerárquico de los módulos que se nota en la figura:



12. Un tanque de agua tiene una capacidad fija en litros que no puede cambiar. Se puede cargar y vaciar un volumen determinado de agua en litros. No se puede llenar más de su capacidad así como tampoco se puede vaciar más del volumen que posee. Se pide implementar un algoritmo que mediante un menú de opciones y de manera modular que permita cargar o descargar una cantidad permitida de agua solicitado por el operador, también se debe poder consultar el volumen de agua en el tanque.

13. Existe una plataforma donde se colocan tanques de agua. Para esta plataforma se quiere implementar un algoritmo denominado: `controlador_de_tanques` que realiza las mismas operaciones que el algoritmo del punto anterior. Al crearse el `controlador_de_tanques` se debe determinar la cantidad máxima de tanques que puede controlar de 2 a 4. El `controlador_de_tanques` permite agregar y quitar tanques de agua. Se desea saber cuántos tanques de agua hay, el volumen ocupado con agua y la capacidad vacía que tiene cada tanque.