**Algoritmos y Programación**

**Unidad 5**

# Ejercicio 5.1. Escribir un programa que permita al usuario ingresar un conjunto de notas, preguntando a cada paso si desea ingresar más notas, e imprimiendo el promedio correspondiente.

def promedio\_notas (n):

“””

Devuelve el promedio de las notas ingresadas.

Parameters:

int or float (n): Notas ingresadas en números enteros o float.

Returns:

(str) Imprime una cadena de texto que indica “El promedio de las notas ingresadas es:”.

“””

assert isinstance (n, int) or (n, float),“Su número ingresado debe ser un número entero o float.”

assert n > 0 and n <= 10, “Su nota debe estar entre 0 y 10.”

notas\_ingresadas = []

notas\_ingresadas.append(n)

mas\_notas = “Si”

suma\_notas = 0

while mas\_notas == “Si”:

n = float(input(“Ingrese otra nota, expresada en números, le diremos su promedio:”))

notas\_ingresadas.append(n)

mas\_notas = input(“¿Quiere ingresar más notas? Si/ No:”))

for ítem in range(len(notas\_ingresadas)):

suma\_notas + = notas\_ingresadas[ítem]

promedio = suma\_notas / len(notas\_ingresadas)

return print(f“El promedio de las notas{notas\_ingresadas} es: {promedio}”)

# Ejercicio 5.3. Manejo de contraseñas:

# Escribir un programa que contenga una contraseña inventada, que le pregunte al usuario la contraseña, y no le permita continuar hasta que la haya ingresado correctamente.

def contrasena ():

“””

Le pregunta al usuario una contraseña y no le permite continuar hasta que no la ingresa correctamente.

Parameters:

(str) Cadena de caracteres y números.

Returns:

(str) Cadena de texto que indica si la contraseña es o no correcta.

“””

password = “2046WKW”

ingreso\_password = input(“Ingrese la contraseña:”)

while not password == ingreso\_password:

ingreso\_password = input(“Su contraseña es incorrecta. Ingrese otra:”)

else:

return print(“Su contraseña es correcta.”)

contrasena()

# Modificar el programa anterior para que solamente permita una cantidad fija de intentos.

def contrasena\_intentos ():

“””

Le pregunta al usuario una contraseña y no le permite continuar hasta que no la ingresa correctamente (se cierra luego de 3 intentos).

Parameters:

(str) Cadena de caracteres y números.

Returns:

(str) Cadena de texto que indica si la contraseña es o no correcta y cuantos intentos le quedan al usuario.

“””

password = “2046WKW”

ingreso\_password = input(“Ingrese la contraseña:”)

intentos = 1

while not password == ingreso\_password and intentos <= 3:

if intentos == 3:

return print(“Contraseña incorrecta. Se acabaron sus intentos.”)

ingreso\_password = input (f“Su contraseña es incorrecta, tiene {3-intentos} intento/s más, ingrese otra contraseña:”)

intentos += 1

else:

return print(“Su contraseña es correcta.”)

contrasena\_intentos()

# Modificar el programa anterior para que después de cada intento agregue una pausa cada vez mayor, utilizando la función sleep del módulo time.

def contrasena\_intentos\_sleep ():

“””

Le pregunta al usuario una contraseña y no le permite continuar hasta que no la ingresa correctamente (se cierra luego de 3 intentos). Se agrega una pausa de tiempo de espera entre cada intento.

Parameters:

(str) Cadena de caracteres y números.

Returns:

(str) Cadena de texto que indica si la contraseña es o no correcta y cuantos intentos le quedan al usuario y cuantos intentos le quedan al usuario.

“””

import time

password = “2046WKW”

ingreso\_password = input(“Ingrese la contraseña:”)

intentos = 1

pausa = 3

while not password == ingreso\_password and intentos <= 3:

if intentos == 3:

return print(“Contraseña incorrecta. Se acabaron sus intentos.”)

print(f“Su contraseña es incorrecta, tiene {3-intentos} intento/s más.”)

time.sleep(pausa)

pausa += 3

ingreso\_password = input (“Ingrese otra contraseña:”)

intentos += 1

else:

return print(“Su contraseña es correcta.”)

contrasena\_intentos\_sleep()

# Modificar el programa anterior para que sea una función que devuelva si el usuario ingresó o no la contraseña correctamente, mediante un valor booleano (True o False).

def contrasena\_bool ():

“””

Le pregunta al usuario una contraseña y no le permite continuar hasta que no la ingresa correctamente (se cierra luego de 3 intentos). Se agrega una pausa de tiempo de espera entre cada intento y una validación True/False.

Parameters:

(str) Cadena de caracteres y números.

Returns:

(bool) “Acceso: True or False”, si la contraseña ingresada fue correcta o no.

(str) Cadena de texto que indica si la contraseña es o no correcta y cuantos intentos le quedan al usuario.

“””

import time

password = “2046WKW”

ingreso\_password = input(“Ingrese la contraseña:”)

intentos = 1

pausa = 3

while not password == ingreso\_password and intentos <= 3:

acceso = False

if intentos == 3:

return print(f“Acceso: {acceso}. Contraseña incorrecta. Se acabaron sus intentos.”)

print(f“Acceso: {acceso}. Su contraseña es incorrecta. Tiene {3-intentos} intento/s más.”)

time.sleep(pausa)

pausa += 3

ingreso\_password = input (“Ingrese otra contraseña:”)

intentos += 1

else:

acceso = True

return print(f“Acceso: {acceso}. “Su contraseña es correcta.”)

contrasena\_bool()

# Ejercicio 5.5. Algoritmo de Euclides:

# Implementar el algoritmo de Euclides para calcular el máximo común divisor de dos números n y m, dado por los siguientes pasos.

# 1. Teniendo n y m, se obtiene r, el resto de la división entera de m/n.

# 2. Si r es cero, n es el mcd de los valores iniciales.

# 3. Se reemplaza m ← n, n ← r, y se vuelve al primer paso.

# Hacer un seguimiento del algoritmo implementado para los siguientes pares de números: (15, 9); (9, 15); (10, 8); (12, 6).

def max\_com\_div (m,n):

“””

Calcula el máximo divisor común entre dos números enteros.

Parameters:

int (m, n): Dos números enteros.

Returns:

(int) El máximo común divisor.

“””

assert isinstance (m, int), “Su número ingresado debe ser entero.”

assert isinstance (n, int), “Su número ingresado debe ser entero.”

r = m % n # r = resto

if r == 0:

return n

else:

while not r == 0:

m = n

n = r

r = m % n

if r == 0:

return n

#Ej.

m = int(input(“Ingrese su 1er número:”))

n = int(input(“Ingrese su 2do número, le diremos el máximo común divisor entre ambos:”))

print(“El máximo común divisor es:” , max\_com\_div(m,n))

# Ejercicio 5.8. Escribir un programa que le pida al usuario que ingrese una sucesión de números naturales (primero uno, luego otro, y así hasta que el usuario ingrese “-1” como condición de salida). Al final, el programa debe imprimir cuántos números fueron ingresados, la suma total de los valores y el promedio.

def num\_centinela (n):

“””

Devuelve cuantos números fueron ingresados, la suma total y el promedio entre ellos.

Parameters:

int or float (n): Números ingresados en números enteros o float.

No puede utilizar el -1, ya que es el valor centinela para salir del programa.

Returns:

(str) Imprime una cadena de texto que indica “La cantidad de números ingresados, la suma total de los mismos y su promedio:”.

“””

assert isinstance (n, int) or (n, float),“Su número ingresado debe ser un número entero o float.”

num\_ingresados = []

num\_ingresados.append(n)

suma\_total = 0

centinela = float(input(“Ingrese un número, (“”-1”” para terminar:”))

while centinela ¡= “-1”:

centinela = float(centinela)

num\_ingresados.append(centinela)

centinela = input(“¿Quiere ingresar otro número? (“”-1”” para terminar):”))

for ítem in range(len(num\_ingresados)):

suma\_total + = num\_ingresados[ítem]

promedio = suma\_total / len(num\_ingresados)

return print(f“La cantidad de números ingresados es: {len(num\_ingresados}, la suma total de ellos es: {suma\_total} y su promedio es: {promedio}”)

# Ejercicio 5.9. Escribir una función que reciba dos números como parámetros, y devuelva cuántos múltiplos del primero hay, que sean menores que el segundo.

# Implementarla utilizando un ciclo for, desde el primer número hasta el segundo.

def multiplos\_for(n1,n2):

“””

Devuelve la cantidad de múltiplos que hay desde el 1er número hasta llegar al segundo exclusive;

los cuenta siempre y cuando su valor sea menor que el 2do número.

Parameters:

int (n1,n2): Dos números enteros.

Returns:

(int) Cantidad de múltiplos desde el 1er número, hasta llegar al 2do número exclusive. Los cuenta siempre y cuando su valor sea menor que el 2do número.

“””

assert isinstance (n1, int),“Su número ingresado debe ser un número entero.”

assert isinstance (n2, int),“Su número ingresado debe ser un número entero.”

multiplos = []

m = 0

for i in range(n1,n2):

m = n1 \* i

if m < n2:

multiplos.append(m)

else:

return(len(multiplos))

#Ej.

n1 = int(input(“Ingrese su 1er número:”))

n2 = int(input(“Ingrese su 2do número, le diremos cuantos múltiplos del 1er número hay entre ellos:”))

print(f“Entre: {n1} y {n2}, hay {multiplos\_for(n1,n2)} múltiplos de : {n1}.”)

# Implementarla utilizando un ciclo while, que multiplique el primer número hasta que sea mayor que el segundo.

def multiplos\_while(n1,n2):

“””

Devuelve la cantidad de múltiplos que hay del 1er número hasta llegar al segundo exclusive;

los cuenta siempre y cuando su valor sea menor que el 2do número.

Parameters:

int (n1,n2): Dos números enteros.

Returns:

(int) Cantidad de múltiplos del 1er número, hasta llegar al 2do número exclusive.

Los cuenta siempre y cuando su valor sea menor que el 2do número.

“””

assert isinstance (n1, int),“Su número ingresado debe ser un número entero.”

assert isinstance (n2, int),“Su número ingresado debe ser un número entero.”

multiplos = []

m = 0

numerador = 1

while m < n2:

m = n1 \* numerador

numerador += 1

if m < n2:

multiplos.append(m)

else:

return(len(multiplos))

#Ej.

n1 = int(input(“Ingrese su 1er número:”))

n2 = int(input(“Ingrese su 2do número, le diremos cuantos múltiplos del 1er número hay hasta el 2do número exclusive :”))

print(f“Hay {multiplos\_for(n1,n2)} múltiplos de : {n1}, hasta {n2} exclusive.”)

# Ejercicio 5.10. Escribir una función que reciba un número natural e imprima todos los números primos que hay hasta ese número.

def numero\_primo(n):

“””

Devuelve verdadero o falso si un número entero es primo o no.

Parameters:

int (n): Números entero mayor a 0.

Returns:

(bool) Verdadero o Falso si el número es primo o no.

“””

assert n> 0,“Su número ingresado debe ser mayor a 0.”

assert isinstance (n, int),“Su número ingresado debe ser un número entero.”

if n < 2:

return False

for i in range(2,n):

if n % i == 0:

return False

else:

return True

def primos(n):

“””

Devuelve todos los números primos que hay entre 1 y el número ingresado inclusive.

Parameters:

int (n): Números entero mayor a 1.

Returns:

(int) Número primos entre 1 y el número ingresado inclusive.

“””

assert n> 1,“Su número ingresado debe ser mayor a 1.”

assert isinstance (n, int), “Su número ingresado debe ser un número entero.”

num\_primos = []

for i in range(2,n+1):

primo = numero\_primo(i) # Le paso la otra función

if primo == True:

num\_primos.append(i)

return num\_primos

#Ej.

n = int(input(“Ingrese su 1er número, le diremos los números primos hasta el ingresado inclusive:”))

print(f“Los números primos hasta {n} son:” ,primos(n))

# Ejercicio 5.12. Escribir una función que dada la cantidad de ejercicios de un examen, y el porcentaje necesario de ejercicios bien resueltos necesario para aprobar dicho examen, revise un grupo de exámenes. Para ello, en cada paso debe preguntar la cantidad de ejercicios resueltos por el alumno, indicando con un valor centinela que no hay más exámenes a revisar. Debe mostrar por pantalla el porcentaje correspondiente a la cantidad de ejercicios resueltos respecto a la cantidad de ejercicios del examen y una leyenda que indique si aprobó o no.

def resultados\_examenes():

“””

Devuelve si el examen está aprobado o no, y el porcentaje correspondiente a la cantidad de

ejercicios resueltos correctamente, respecto de la cantidad de ejercicios del examen.

Parameters:

(int, float) Para los datos numéricos.

(str) El valor centinela es (\*)

Returns:

Print(“Examen aprobado con: % / 100 o Examen no aprobado.”)

“””

ejercicios = int(input(“Indique la cantidad de ejercicios que tiene su examen:”))

aprobar = float(input(“Indique la cantidad de ejercicios que deben estar correctos para aprobar su examen:”))

ej\_correctos = input(“¿Cuántos ejercicios resolvió correctamente este alumno? (\*para terminar):”)

while not ej\_correctos == “\*”:

ej\_correctos = float(ej\_correctos)

if not ej\_correctos >= aprobar:

print(“Examen no aprobado.”)

else:

porcentaje = (ej\_correctos \* 100) / ejercicios

print(f“Examen aprobado con: {porcentaje} / 100”)

ej\_correctos = input(“¿Quiere ingresar otro examen? ¿Cuántos ejercicios resolvió correctamente este alumno? (\*para terminar):”)

resultados\_examenes()

**Otra respuesta con nombre de alumnos, cantidad de aprobados y porcentaje de aprobados:**

<https://github.com/Plofi/Tareas/blob/master/Clase%204/Algoritmos%20y%20Programaci%C3%B3n%20-%20Unidad%205/AyP_Unidad%205_Ej%205.12_Def%20examenes_alumnos.py>