**LABORATORIO**

**Hora prevista**

15-20 minutos

**Nivel de dificultad**

Medio

**Prerrequisitos**

LAB Listas y return

**Objetivos**

Familiarizar al alumno con:

* proyectar y escribir funciones parametrizadas;
* utilizando la declaración de return;
* utilizando las funciones propias del alumno.

**Guión**

Su tarea es escribir y probar una función que toma dos argumentos (un año y un mes) y devuelve el número de días para el par mes / año dado (aunque solo febrero es sensible al valor del año, su función debería ser universal).

La parte inicial de la función está lista. Ahora, modifique a la función para que use la opción de return  None si sus argumentos no tienen sentido.

Por supuesto, puede (y debe) usar la función previamente escrita y probada (LAB Listas y return). Puede ser de mucha ayuda. Lo alentamos a que use una lista con los meses. Puede crearlo dentro de la función: este truco acortará significativamente el código.

Hemos preparado un código de prueba.

def isYearLeap(year):

#

# Codigo from LAB Listas y return

#

def daysInMonth(year, month):

#

# put your new code here

#

testYears = [1900, 2000, 2016, 1987]

testMonths = [2, 2, 1, 11]

testResults = [28, 29, 31, 30]

for i in range(len(testYears)):

yr = testYears[i]

mo = testMonths[i]

print(yr, mo, "->", end="")

result = daysInMonth(yr, mo)

if result == testResults[i]:

print("OK")

else:

print("Failed")

Resolución:

def es\_bisiesto(anio: int) -> bool:

return anio % 4 == 0 and (anio % 100 != 0 or anio % 400 == 0)

def obtener\_dias\_del\_mes(mes: int, anio: int) -> int:

if mes in [4, 6, 9, 11]:

return 30

if mes == 2:

if es\_bisiesto(anio):

return 29

else:

return 28

else:

return 31

mes=int(input("Ingrese el mes: "))

if mes>12 or mes<1:

raise Exception("El mes ingresado no existe")

anio = int( input("Ingrese el año: "))

dias = obtener\_dias\_del\_mes(mes, anio)

print("El mes ", mes," en el año ",anio," tiene ",dias," días")