**LABORATORIO Listas y return**

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):**

**if year % 4 == 0:**

**if year % 100 == 0:**

**if year % 400 == 0:**

**return True**

**else:**

**return False**

**else:**

**return True**

**else:**

**return False**

**def daysInMonth (year, month):**

**Meses = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]**

**#Valida meses coherentes**

**if month in Meses:**

**if month == 4 or month == 6 or month == 9 or month == 11:**

**return 30**

**elif month == 2:**

**#Febrero 29 dias**

**if isYearLeap(year) == True:**

**return 29**

**#Febrero 28 dias**

**else:**

**return 28**

**else:**

**return 31**

**else:**

**return None**

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")

**Desarrollo del ejercicio:**

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente