



UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO



FACULTAD DE ESTUDIOS
SUPERIORES ARAGÓN

Ingeniería en Computación

DISEÑO Y ANALISIS DE ALGORITMOS



Tarea 6 ejercicios de problemas que pueden ser resueltos
con aritmética modular

Profesor: Marcelo Pérez Medel

Leonardo Olvera Martínez

Grupo: 1507

Fecha: martes 07 de noviembre de 2023

Realice los siguientes problemas

1.- Realice un programa que reciba un valor de 0 a 6 que corresponden a los días de la semana, es decir, “domingo”, “lunes”, “martes”, “miércoles”, “jueves”, “viernes” y “sábado” y después un valor numérico que corresponderá a la cantidad de días posteriores y devuelva el día de la semana que será después de sumar ese valor, por ejemplo, si escribimos:

Print(calcula_dia(0, 1))

Debe imprimir:

“lunes”

Porque el primer parámetro que le pasamos es 0 que representa el domingo y el segundo día que le pasamos es 1, así que 1 día después del domingo es “lunes”

Print(calcula_dia(0, 2))

Debe imprimir:

“martes”

Print(calcula_dia(5, 2))

Debe imprimir:

“lunes”

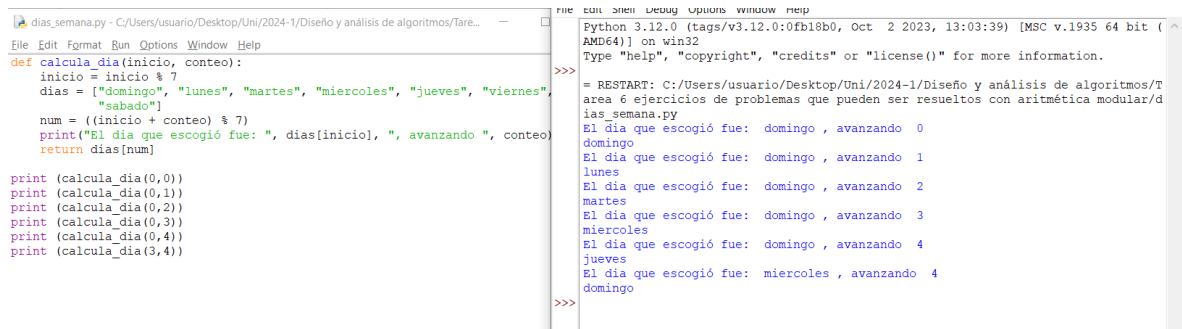
El código que sea realizó fue el siguiente:

```
def calcula_dia(inicio, conteo):  
    inicio = inicio % 7  
    dias = ["domingo", "lunes", "martes", "miercoles", "jueves", "viernes",  
            "sabado"]  
    num = ((inicio + conteo) % 7)  
    print("El día que escogió fue: ", dias[inicio], ", avanzando ", conteo)  
    return dias[num]
```

Primero definimos una función en donde recibamos 2 números, el primero, inicio, en donde recibiremos un numero en donde empezaremos a contar, el segundo es conteo, el cual comenzará a contar los días a partir de “inicio”.

Después, usando %, calculamos el módulo del inicio (esto por si ingresamos un dato mayor a la cantidad de días que tenemos en la semana), a continuación, creamos un arreglo que contenga los días de la semana y finalmente obtenemos el módulo

de la suma entre el inicio y el conteo para, posteriormente, buscar en el arreglo, el índice de nuestro residuo.



```
dias_semana.py - C:/Users/usuario/Desktop/Uni/2024-1/Diseño y análisis de algoritmos/Tare...
File Edit Format Run Options Window Help
def calcula_dia(inicio, conteo):
    inicio = inicio % 7
    dias = ["domingo", "lunes", "martes", "miercoles", "jueves", "viernes",
            "sabado"]
    num = (inicio + conteo) % 7
    print("El día que escogió fue: ", dias[inicio], ", avanzando ", conteo)
    return dias[num]

print(calcula_dia(0,0))
print(calcula_dia(0,1))
print(calcula_dia(0,2))
print(calcula_dia(0,3))
print(calcula_dia(0,4))
print(calcula_dia(3,4))

Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/usuario/Desktop/Uni/2024-1/Diseño y análisis de algoritmos/Tarea 6 ejercicios de problemas que pueden ser resueltos con aritmética modular/dias_semana.py
El día que escogió fue: domingo , avanzando 0
domingo
El día que escogió fue: domingo , avanzando 1
lunes
El día que escogió fue: domingo , avanzando 2
martes
El día que escogió fue: domingo , avanzando 3
miercoles
El día que escogió fue: domingo , avanzando 4
jueves
El día que escogió fue: miercoles , avanzando 4
domingo
>>>
```

2.- Verificación de año bisiesto, Un año es bisiesto si es divisible por 4, excepto los años que son divisibles por 100 pero no por 400. Implementa una función que verifique si un año es bisiesto utilizando operaciones modulares. Es decir, el año 1996 fue bisiesto porque es divisible entre 4. 1900 es divisible entre 4, pero al ser también entre 100 no es bisiesto, así 1896 fue bisiesto y el siguiente fue hasta 1904, ahí pasaron 8 años para tener un bisiesto, al saltarse el 1900. El año 2000 es divisible entre 4 por lo que sería bisiesto, pero al ser divisible entre 100 no debería serlo, pero como la excepción es que sea divisible entre 400 y 2000 lo es, si lo fue.

El código que resuelve el siguiente problema es este:

```
def bisiesto(anio):
    cuatro = anio % 4
    cien = anio % 100
    cuatrocientos = anio % 400
    dosmil = anio % 2000
    if(cuatro == 0 and cien != 0) or (cuatro == 0 and cien == 0 and
                                     cuatrocientos == 0 and dosmil == 0):
        return "Es bisiesto"
    else:
        return "No es bisiesto"
```

Primero definimos una función la cual pedirá el año para poder comprobar, posteriormente creamos 4 variables en donde obtengamos el módulo de 4 y de 100, para comprobar que si es divisible entre 4 pero no entre 100 es un bisiesto, esto lo comprobaremos con un if y le agregaremos otra opción en donde obtengamos el módulo de cuatrocientos y dos mil, en donde veremos si son divisibles entre nuestras 4 anteriores variables también es bisiesto, en caso de no ser ninguna de estas dos opciones, no es bisiesto.

IDLE Shell 3.12.0

File Edit Shell Debug Options Window Help

Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [AMD64] on win32

Type "help", "copyright", "credits" or "license()" for more in

>>>

= RESTART: C:/Users/usuario/Desktop/Uni/2024-1/Diseño y análisis
area 6 ejercicios de problemas que pueden ser resueltos con ar

iciesto.py
Es biciesto
No es biciesto
Es biciesto
Es biciesto
No es biciesto
Es biciesto
Es biciesto
>>>

biciesto.py - C:/Users/usuario/Desktop/Uni/2024-1/Diseño y análisis de algoritmos/Tarea 6 e...

File Edit Format Run Options Window Help

```
def biciesto(anio):  
    cuatro = anio % 4  
    cien = anio % 100  
    cuatrocientos = anio % 400  
    dosmil = anio % 2000  
    if (cuatro == 0 and cien != 0) or (cuatip == 0 and cien == 0 and  
        cuatrocientos == 0 and dosmil == 0):  
        return "Es biciesto"  
    else:  
        return "No es biciesto"  
  
print(bisiesto(1904))  
print(bisiesto(1900))  
print(bisiesto(1908))  
print(bisiesto(1996))  
print(bisiesto(2000))  
print(bisiesto(2001))  
print(bisiesto(1904))  
print(bisiesto(2004))
```