Relatoria 2

El siguiente es un documento con el resumen de las clases del segundo corte, en el cual vimos temas como métodos de strings, métodos de listas y métodos de diccionarios. También veremos algunos ejemplos vistos en clase y se tratará de ampliar más la definición de algunos conceptos.

CLASE 1:
Métodos de strings

Hay dos opciones para encontrar un string dentro de una string en Python, find() y rfind().

Cada uno retorna la posición en la que se encuentre la substring. La diferencia entre los dos, es que find() retorna la posición de la primera similitud de la substring y rfind() retorna la última posición de la similitud de la substring.

Nosotros podemos proveer algunos argumentos opcionales de inicio y fin para limitar la búsqueda de la substring, dentro de los límites del string.

Ejemplo:

```
# Estan retornando una nueva cadena sin alterar la original

# Cadenas son inmutables (no se pueden alterar)

#---------------------- Indexación y slicing -------------------------

# Indexacion: Es un método para acceder a los elementos de las cadenas

# slicing: Es un método para acceder a rebanadas de las cadenas

cadena2 = "Mensaje para Informatica 3"

# Indexacion

print("primer => ", cadena2[0]) #primer elemento de la cadena

print("Segundo => ", cadena2[1]) #Segundo elemento de la cadena

print("PenUltimo => ", cadena2[-2]) #PenUltimo elemento de la cadena
```

```
print("Ultimo => ", cadena2[-1]) #Ultimo elemento de la cadena

#Slicing

#Extraer toda la cadena al revés

print("Opcion 1: ", cadena2[-1:-27:-1])

print("Opcion 2: ", cadena2[-1::-1])

print("Opcion 3: ", cadena2[::-1])

#Extraer cada tercer elemento empezando en el segundo indice

print("Opcion 1: ", cadena2[2:27:3])

print("Opcion 2: ", cadena2[2::3])

print("Opcion 3: ", cadena2[-24::3])

#Extraer los dos elementos en la mitad de la cadena

print("Opcion 1: ", cadena2[12:14:1])

print("Opcion 2: ", cadena2[-14:-12:1])
```

CLASE 2:
Métodos de listas:

Un método es lo mismo que una función, excepto que se «llama» a un valor. Por ejemplo, si una lista fuera almacenado en spam, usted llamaría al método de lista index() (que se explicará en breve) en esa lista así: spam.index('hello'). **La parte del método viene después del valor**, separada por un punto.

Cada tipo de datos tiene su propio conjunto de métodos. El tipo de datos lista, por ejemplo, tiene varios métodos útiles para **encontrar, añadir, eliminar y manipular** valores en una lista.

*Haga una copia de lista3 de la siguiente manera

lista3Copia = lista3

y agregue 3 nuevos elementos sobre esta nueva lista.

¿qué sucede con la lista3 original?

```
""")
lista3Copia = lista3 # Esto no es una verdadera copia, esto es un nuevo apodo a
# a la lista3 original
lista3Copia.append("Cristian")
lista3Copia.append("Elias")
lista3Copia.append("Pachon")
print(lista3, "original sin afectar \"supuestamente\"")
print(lista3Copia, "copia afectada \"solamente\"")
lista1 = [1,2,3,4,5,6,7,8]
lista2 = list(range(20)) + ["a", "b", "c", 100]
lista3 = ["cruel", "mundo", "hola", 1, 100, 200, 500]
print("Extraer el elemento inicial de lista1")
print(lista1[0], lista1[-8])
print("Extraer el elemento final de lista1 (de dos maneras)")
print(lista1[7], lista1[-1])
print("Extraer el elemento del medio de lista2 (de dos maneras)")
print(lista2[11], lista2[-13])
print("""Extraer los primeros 3 elementos de lista1, lista2 y lista3
y colocarlos en una nuevaLista""")
print(lista1[0:3] + lista2[0:3] +lista3[0:3])
print("Extraer cada 2 elementos de lista 2")
print(lista2[::2])
print("Extraer todos los elementos de lista3 al revés")
print(lista3[::-1])
print("Extraer los elementos de lista3 ubicados en indices impares")
print(lista3[1::2])
print(lista3)
```

CLASE 3:
Modulos y paquetes:

Módulos

Un *módulo* es un archivo de Python cuyos objetos (funciones, clases, excepciones, etc.) pueden ser accedidos desde otro archivo. Se trata simplemente de una forma de organizar grandes códigos.

Consideremos, por ejemplo, un archivo aritmetica.py que contenga las siguientes definiciones.

```
def sumar(a, b):
    return a + b

def restar(a, b):
    return a - b

def mult(a, b):
    return a * b

def div(a, b):
    return a / b
```

Paquetes

Un *paquete* es una carpeta que contiene varios módulos. Siguiendo el ejemplo anterior, podemos diseñar un paquete matematica creando una carpeta con la siguiente estructura.

```
matematica/
    |-- __init__.py
    |-- aritmetica.py
    |-- geometria.py
```

Debe contener siempre un archivo __init__.py (por el momento vacío) para que Python entienda que se trata de un paquete y no de una simple carpeta. Así, podemos acceder a alguno de los módulos del paquete de la siguiente manera.

import matematica.aritmetica

print(matematica.aritmetica.sumar(7, 5))

O bien de la siguiente.

from matematica **import** aritmetica

print(aritmetica.sumar(7, 5))

Ejemplo con fechas:

```
"""
Este modulo contiene fechas de interes
para el profe
"""
fechaImportante1 = "20-07" # Dia independencia
fechaImportante2 = "15-05" # Dia del profe
fechaImportante3 = "06-03" # Nacimiento Gabriel Garcia Marquez
fechaImportante4 = "12-08" # Adopte un gatico
fechaImportante5 = "31-12" # Fin de año
fechaImportante6 = "31-06" # Mitad de año
```

Ejemplo con interfaz:

```
"""
Este modulo sirve para imprimir unos mensajes,
este modulo no almacena nada
"""
```

```
def imprimirMensaje(nombre):
mensaje = "hola, soy holaMundo otra vez" + nombre
print(mensaje)
return None
def imprimirSeparador(tipo):
print("\n" + tipo * 50 + "\n")
def imprimirVariable(nombre, variable):
print(nombre + " ==> " + str(variable))
def imprimirListado(lista):
for elemento in lista:
imprimirSeparador("-")
print(elemento)
```

Ejemplo con Lógica:

```
Este modulo sirve para realizar algunas operaciones
y se almacenan algunas variables
contiene dos funciones
"""
def sumar2Nums(numero1, numero2):
resultado = numero1 + numero2
return resultado
def sumarNNumeros(*numeros): #numeros se interpreta como un listado
suma = sum(numeros)
return suma
##### Para hacer el testeo del modulos ####
```

```
if __name__ == "__main__":
a = sumar2Nums(1,2)
print(a)
b = sumarNNumeros(1,2,3,4,5,6,7,8)
print(b)
```

Ejemplo con Main:

```
"""""""""
Crear un modulo llamado fechas en el cual almacene fechas importantes
ejemplo:
fechaImportante1 = "20-07"
fechaImportante2 = "15-05"
fechaImportante3 = "06-03"
... 3 mas
consuma desde main.py todas las fechas importantes,
almacenandolas en una lista llamada listaFechasImportantes
imprima cada elemento del listado
separando las fechas consumiendo la funcion imprimirSeparador
"""""
import fechas
f1 = fechas.fechaImportante1
f2 = fechas.fechaImportante2
f3 = fechas.fechaImportante3
f4 = fechas.fechaImportante4
f5 = fechas.fechaImportante5
f6 = fechas.fechaImportante6
listaFechasImportantes = [f1, f2, f3, f4, f5, f6]
```

```
interfaz.imprimirListado(listaFechasImportantes)
```