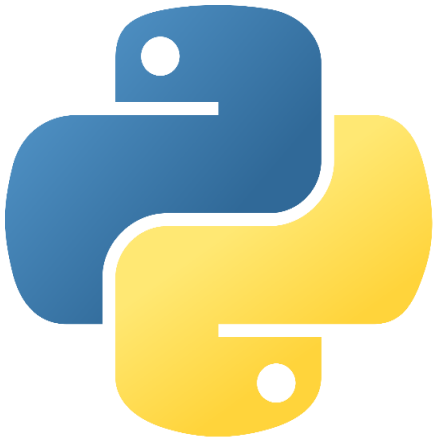

EJERCICIOS DE PYTHON II

Cadenas y Listas

Cadenas:

- Ingreso de datos
- Métodos para validar
- Formateo de texto e impresión
- Replicación
- Manipulación
- Segmentación
- Texto multilínea

Listas:

- Creación y Manipulación. Acceso por subíndice
- Métodos para agregar, buscar, modificar y eliminar elementos
- Recorrer con for, while y for-in
- Concatenación
- Desempaquetado
- Funciones para obtener el largo, sumar elementos, buscar el máximo y el mínimo
- Operadores de pertenencia
- Funciones de ordenamiento

Codo a Codo
2024

1) **Informe de Datos Estudiantiles:** La universidad requiere un programa capaz de generar un informe detallado con los datos del estudiante, su nombre de usuario, domicilio, correo electrónico y contraseña. Debes seguir las siguientes pautas:

   Ingreso de datos:

   - El usuario debe ingresar su dni, nombre, apellido y domicilio, atendiendo a las siguientes restricciones:
     a) DNI: Debe contener solamente números.
     b) Nombre: No puede tener más de 30 caracteres.
     c) Apellido: Debe contener únicamente letras y un solo apellido.
     d) Domicilio: Puede contener cualquier combinación de letras y números en la dirección de calle, pero en el número debe haber al menos un número presente y no tener caracteres alfabéticos.

   Formato del informe:

   - La línea 1 del informe debe contener una fecha alineada a la derecha.
   - Las líneas 2 y 4 serán rellenadas con 56 caracteres "=", para garantizar el formato adecuado.
   - La línea 3 contendrá el string “universidad de python - datos del estudiante”, centrado y con el formato “tipo título” que se ve en el modelo.
   - A continuación, informar los datos siguiendo estas indicaciones:
     a) ID: Se genera a partir del DNI y se presenta con una longitud de 11 caracteres, llenando con ceros a la izquierda si es necesario.
     b) NOMBRE COMPLETO: El apellido debe estar en mayúsculas, se agrega una coma, un espacio y el nombre que debe estar en formato de título (primera letra de cada palabra en mayúscula).
     c) DOMICILIO: Se presenta en mayúsculas (calle), un espacio y el número.
     d) USUARIO: El apellido se presenta en minúsculas, se agrega un guion y la primera inicial del nombre en minúsculas, finalmente se acompaña con los primeros tres dígitos del DNI.
     e) CORREO ELECTRÓNICO: El apellido se presenta en minúsculas, se agrega un punto y el primer nombre* en minúsculas. Al final se agrega “@unipython.com.ar”.
        * En el caso de nombres compuestos transformar el string en una lista y conservar el primer elemento
     f) CONTRASEÑA: La primera letra del apellido se presenta en mayúscula, sin dejar espacio se agrega la primera letra del nombre en minúscula, un guion y los últimos 3 dígitos del DNI.
   - Dejar una línea en blanco luego de la contraseña y mostrar en 3 líneas utilizando triple comilla simple lo siguiente:

     Recomendamos cambiar su contraseña.
     La presente información es confidencial.
     Dpto. de Sistemas.

   Nota: Se recomienda seguir el modelo que se acompaña al final de ejercicio y atender a las indicaciones.

Salida por pantalla:

```
                                                        31/10/2023
============================================================
===== Universidad De Python - Datos Del Estudiante =====
============================================================
ID: 00034521879
NOMBRE COMPLETO: SANCHEZ, Diego Alejandro
DOMICILIO: AV. CORRIENTES 2648
USUARIO: sanchez-d345
CORREO ELECTRÓNICO: sanchez.diego@unipython.com.ar
CONTRASEÑA: Sd-879

Recomendamos cambiar su contraseña.
La presente información es confidencial.
Dpto. de Sistemas.
```

```python
# Validaciones de datos
# a) DNI: Debe contener solamente números.
dni = input("Ingrese su DNI: ")
while not dni.isdigit(): # Se verifica si el DNI contiene solamente dígitos
     dni = input("Debe contener solamente números. Ingrese su DNI: ")

# b) Nombre: No puede tener más de 30 caracteres.
nombre = input("Ingrese su nombre: ")
while len(nombre) > 30: # Se verifica si el nombre tiene más de 30 caracteres
    nombre = input(f'{nombre}, su nombre es muy largo! ({len(nombre)}) Ingrese un nombre más corto: ')

#c) Apellido: Debe contener únicamente letras.
apellido = input("Ingrese su apellido: ")
while not apellido.isalpha() or ' ' in apellido: # Se verifica si el apellido contiene únicamente letras y no contiene espacios
    apellido = input("El apellido debe contener únicamente letras y no debe contener espacios. Ingrese su apellido: ")

#d) Domicilio: Puede contener cualquier combinación de letras y números en la dirección de calle, pero en el número debe haber al menos un número presente y no tener caracteres alfabéticos.
domicilio_calle = input("Ingrese su domicilio (calle): ")
domicilio_numero = input("Ingrese su domicilio (numero): ")
digitos = 0
caracteres = 0

for caracter in domicilio_numero: # Se cuenta la cantidad de dígitos y caracteres alfabéticos en el número del domicilio
    if caracter.isdigit():
        digitos += 1
```

```
    if caracter.isalpha():
        caracteres += 1

while digitos == 0 or caracteres > 0:  # Se solicita al usuario ingresar
un número de domicilio válido
    domicilio_numero = input("Debe contener al menos un número y no puede
contener letras. Ingrese su domicilio (numero): ")
    digitos = 0
    caracteres = 0
    for caracter in domicilio_numero:
        if caracter.isdigit():
            digitos += 1
        if caracter.isalpha():
            caracteres += 1

# Datos para impresión
linea = "="*56
encab = ' universidad de python - datos del estudiante '
titulo = encab.title()
titulo_centrado = titulo.center(56, "=")
fecha = '31/10/2023'
fecha = fecha.rjust(56)

# Informe
print(fecha)
print(linea)
print(titulo_centrado)
print(linea)

usuario = dni.zfill(11)
print(f'ID: {usuario}')
print(f'NOMBRE COMPLETO: {apellido.upper()}, {nombre.title()}')
print(f'DOMICILIO: {domicilio_calle.upper()} {domicilio_numero}')
print(f'USUARIO: {apellido.lower()}-{nombre[0].lower()}{dni[:3]}')
nombres = nombre.split(' ')
print(f'CORREO ELECTRÓNICO:
{apellido.lower()}.{nombres[0].lower()}@unipython.com.ar')
print(f'CONTRASEÑA: {apellido[0].upper()}{nombre[0].lower()}-{dni[-3:]}')
# empezando desde el tercer dígito contando desde el final hasta el
último caracter

leyenda = '''
Recomendamos cambiar su contraseña.
La presente información es confidencial.
Dpto. de Sistemas.
'''
print(leyenda)
```

2) **Gestión de Materias:** La universidad necesita un programa para gestionar sus cursos, estudiantes y notas. Implementar las siguientes funcionalidades utilizando listas:
   a) Crear una lista llamada Materias que contendrá 3 materias (harcodeado). Mostrar en pantalla.
   b) Agregar una materia al final y otra en la segunda posición. Mostrar en pantalla la lista, la posición de la primera materia y la posición de la última.
   c) Solicitar una materia por teclado. Verificar si existe y en caso de que no exista agregarla al final de la lista, recorrer la lista utilizando un bucle ***for***. Mostrar la lista en pantalla y un mensaje de confirmación.
   d) Solicitar una materia por teclado y modificar su nombre por otro ingresado por teclado. Realizar la búsqueda verificando si la materia existe utilizando un bucle ***while***. Mostrar la lista en pantalla y un mensaje de confirmación.
   e) Solicitar una materia por teclado y eliminarla. Realizar la búsqueda verificando si la materia existe utilizando un bucle ***for - in***. Mostrar la lista en pantalla sólo si la materia fue eliminada, junto con un mensaje de confirmación.
   f) Eliminar la última materia de la lista y la primera materia de la lista. En ambos casos verificar que la lista no esté vacía.

   Nota: Las operaciones de este ejercicio están relacionadas con lo que se conoce como un sistema CRUD (Create, Read, Update, Delete) en el contexto de la gestión de datos:
   - **Create:**
     b) Agregar una materia al final y otra en la segunda posición.
     c) Solicitar una materia por teclado. Verificar si existe y agregarla al final si no existe.
   - **Read (Leer):**
     a) Mostrar la lista de materias.
     Update (Actualizar):
     d) Solicitar una materia por teclado y modificar su nombre por otro ingresado por teclado.
   - **Delete (Eliminar):**
     e) Solicitar una materia por teclado y eliminarla.
     f) Eliminar la última materia de la lista y la primera materia de la lista.

   Todas estas operaciones forman parte de un conjunto completo de acciones que permiten crear, leer, actualizar y eliminar datos en una lista de materias.

Posible salida por pantalla:

```
['Python I', 'Bases de datos I', 'Filosofía']
Lista: ['Python I', 'Python II', 'Bases de datos I',
'Filosofía', 'Pseudocódigo']
Primera materia: Python I
Última materia: Pseudocódigo
['Python I', 'Python II', 'Bases de datos I',
'Filosofía', 'Pseudocódigo']
Ingrese el nombre de la materia: Python III
Materia Python III agregada.
Lista actualizada: ['Python I', 'Python II', 'Bases de
datos I', 'Filosofía', 'Pseudocódigo', 'Python III']
Ingrese el nombre de la materia a modificar: Python I
Ingrese el nuevo nombre para la materia: Introd. a
Python
Materia Python I modificada a Introd. a Python.
```

```
Ingrese el nombre de la materia a eliminar: Pseudocódigo
Materia Pseudocódigo eliminada.
Lista actualizada de materias: ['Introd. a Python',
'Python II', 'Bases de datos I', 'Filosofía', 'Python
III']Se eliminó la última materia: Python III
Se eliminó la primera materia: Introd. a Python
```

```python
# a) Crear una lista llamada Materias que contendrá 3 materias
(harcodeado). Mostrar en pantalla.
materias = ["Python I", "Bases de datos I", "Filosofía"]
print(materias)

# b) Agregar una materia al final y otra en la segunda posición. Mostrar
en pantalla la lista, la posición de la primera materia y la posición de
la última.
materias.append("Pseudocódigo")
materias.insert(1, "Python II")
print(f'Lista: {materias}')
print(f'Primera materia: {materias[0]}')
print(f'Última materia: {materias[len(materias)-1]}')

# c) Solicitar una materia por teclado. Verificar si existe y en caso de
que no exista agregarla al final de la lista, recorrer la lista
utilizando un bucle for. Mostrar la lista en pantalla y un mensaje de
confirmación.
mat = input("Ingrese el nombre de la materia: ")
encontrado = False
for m in range(len(materias)):
    if materias[m].upper() == mat.upper():
        encontrado = True
if encontrado == False:
    materias.append(mat)
    print(f'Materia {mat} agregada.')
else:
    print(f'No es posible sumar la materia {mat}, ya existe.')
print(f'Lista actualizada: {materias}')

# d) Solicitar una materia por teclado y modificar su nombre por otro
ingresado por teclado. Realizar la búsqueda verificando si la materia
existe utilizando un bucle while. Mostrar la lista en pantalla y un
mensaje de confirmación.
mat = input("Ingrese el nombre de la materia a modificar: ")
nuevo_nombre = input("Ingrese el nuevo nombre para la materia: ")
encontrado = False
indice = 0
while indice < len(materias) and not encontrado:
    if materias[indice].upper() == mat.upper():
        encontrado = True
        materias[indice] = nuevo_nombre
```

```
    indice += 1
if encontrado:
    print(f'Materia {mat} modificada a {nuevo_nombre}.')
else:
    print(f'La materia {mat} no existe en la lista.')

# e) Solicitar una materia por teclado y eliminarla. Realizar la búsqueda
verificando si la materia existe utilizando un bucle for - in. Mostrar la
lista en pantalla sólo si la materia fue eliminada, junto con un mensaje
de confirmación.
mat = input("Ingrese el nombre de la materia a eliminar: ")
encontrado = False

for materia in materias:
    if materia.upper() == mat.upper():
        materias.remove(materia)
        encontrado = True

if encontrado:
    print(f'Materia {mat} eliminada.')
    print(f'Lista actualizada de materias: {materias}')
else:
    print(f'La materia {mat} no existe en la lista.')

#f) Eliminar la última materia de la lista y la primera materia de la
lista.
# Eliminar la última materia de la lista
if len(materias) > 0:
    ultima_materia = materias.pop()
    print(f'Se eliminó la última materia: {ultima_materia}')
else:
    print('No hay materias en la lista.')

# Eliminar la primera materia de la lista
if len(materias) > 0:
    primera_materia = materias.pop(0)
    print(f'Se eliminó la primera materia: {primera_materia}')
else:
    print('No hay materias en la lista.')

print(f'Lista actualizada de materias: {materias}')
```

3) **Estadísticas sobre edades de alumnos:**
   a) Crear dos listas que contengan edades de un grupo de estudiantes, llamarlas **lista1** y **lista2**. Concatenarlas en una lista llamada **edades**. Mostrar las tres listas.
   b) Determinar la edad más alta y mostrarla en pantalla.
   c) Determinar la edad más baja y mostrarla en pantalla.
   d) Obtener la posición de la edad más baja.
   e) Obtener el promedio de las edades.

f) Contar cuántos estudiantes tienen alguna edad que ***no*** se encuentre en la lista.
g) Contar cuántos estudiantes tienen alguna edad que se encuentre en la lista.
h) Invertir el orden de las edades y mostrarlas en pantalla.
i) Ordenar las edades de menor a mayor y mostrarlas en pantalla.
j) Ordenar las edades de mayor a menor y mostrarlas en pantalla.
k) Borrar la lista de edades y mostrarla en pantalla.

Extra: Desempaquetado y pertenencia

l) Crear una lista llamada turnos que contenga los valores Mañana, Tarde y Vespertino. Verificar la pertenencia de "Noche" en la lista y "Mañana" en la lista. Imprimir los resultados.
m) Desempaquetar la lista y mostrar los resultados individuales.

Salida por pantalla:

```
Primera lista: [26, 38, 47, 30, 50, 19, 50, 48]
Segunda lista: [35, 40, 45, 55, 21]
Listas unidas: [26, 38, 47, 30, 50, 19, 50, 48, 35, 40,
45, 55, 21]
Edad más alta: 55
Edad más baja: 19
Posición de la edad más baja: 5
Promedio de edades: 38.77
Cantidad de 18 años: 0
Cantidad de 50 años: 2
Edades invertidas: [21, 55, 45, 40, 35, 48, 50, 19, 50,
30, 47, 38, 26]
Edades ordenadas: [19, 21, 26, 30, 35, 38, 40, 45, 47,
48, 50, 50, 55]
Edades ordenadas al revés: [55, 50, 50, 48, 47, 45, 40,
38, 35, 30, 26, 21, 19]
Lista vacía: []
Lista de turnos: ['Mañana', 'Tarde', 'Vespertino']
Noche está en la lista? False
Mañana está en la lista? True
Turno 1: Mañana
Turno 2: Tarde
Turno 3: Vespertino
```

```python
# Estadísticas sobre edades de alumnos:
lista1 = [26,38,47,30,50,19,50,48]
lista2 = [35,40,45,55,21]
edades = lista1 + lista2
print(f'Primera lista: {lista1}')
print(f'Segunda lista: {lista2}')
print(f'Listas unidas: {edades}')
print(f'Edad más alta: {max(edades)}')
print(f'Edad más baja: {min(edades)}')
print(f'Posición de la edad más baja: {edades.index(min(edades))}')
print(f'Promedio de edades: {sum(edades)/len(edades):.2f}')
print(f'Cantidad de 18 años: {edades.count(18)}')
print(f'Cantidad de 50 años: {edades.count(50)}')
edades.reverse()
print(f'Edades invertidas: {edades}')
edades.sort()
print(f'Edades ordenadas: {edades}')
edades.sort(reverse=True)
print(f'Edades ordenadas al revés: {edades}')
edades.clear()
print(f'Lista vacía: {edades}')

# Extra: Desempaquetado y pertenencia
turnos = ["Mañana", "Tarde", "Vespertino"]
print(f'Lista de turnos: {turnos}')
print(f'Noche está en la lista? {"Noche" in turnos}')
print(f'Mañana está en la lista? {"Mañana" in turnos}')
turno1, turno2, turno3 = turnos
print(f'Turno 1: {turno1}')
print(f'Turno 2: {turno2}')
print(f'Turno 3: {turno3}')
```