Facultad de Ciencias
UNER de la Administración

TRABAJO PRÁCTICO Nº 1 – INTRODUCCIÓN A PYTHON

Unidades 1 y 2 – POO y Python

PROGRAMACIÓN 2 - 2022 – 2do cuatrimestre

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

EL TRABAJO PRÁCTICO Nº 1 TIENE POR OBJETIVO QUE EL ALUMNO

- Repase los conocimientos adquiridos de Python desarrollados durante el cursado de Programación 1.
- Se introduzca en los conceptos fundamentales y relacionados a la Programación Orientada a Objetos.

CONDICIONES DE ENTREGA

- El Trabajo Práctico deberá ser:
    - Realizado en forma individual o en **grupos de NO más de 4 (cuatro) alumnos**.
    - Cargado en la sección del Campus Virtual correspondiente, en un archivo ZIP o RAR con las soluciones a cada ejercicio. Cada solución debe estar contenida en un archivo .py distinto.
    - En caso de realizar el Trabajo Práctico en grupo, deberá indicarse el apellido y nombre de los integrantes del mismo. Todos los integrantes del grupo deben realizar la entrega en el campus y deberá agregarse al comprimido con las soluciones un archivo *integrantes.txt* con los nombres de los participantes.
    - Entregado antes de la fecha límite informada en el campus.
- El Trabajo Práctico será calificado como Aprobado o Desaprobado.
- Las soluciones del alumno/grupo deben ser de autoría propia. De encontrarse soluciones idénticas entre diferentes grupos, dichos trabajos prácticos serán clasificados como **DESAPROBADO**, lo cual será comunicado en la devolución.

EJERCICIOS:

1. **Sección A: Ejercicios básicos de Python**

   Resolver cada ejercicio en un archivo diferente.

   a. Escribir una función de nombre *palabra_no_tiene_letras(palabra, letras_prohibidas)*, la cual retorne True si es que los caracteres que componen una palabra no se encuentran en una lista de caracteres prohibidos.

   b. Escribir una función de nombre *es_abc(palabra)* la cual retorne True siempre y cuando las letras que componen dicha palabra estén en orden alfabético, y False en caso contrario.

   c. Escriba un procedimiento *procesar_palabras(entrada)* que acepte una secuencia de palabras separadas por coma, las ordene y las imprima. Suponiendo que la entrada provista al programa es la siguiente:
   `te,felicito,que,bien,actuas`
   La salida esperada es:
   `actuas,bien,felicito,que,te`

   d. Dadas dos listas, lista1 y lista2, escribir un método *listas_diferencia(lista1, lista2)* que tome ambas como parámetros e imprima dos listas, cada una con:
      i. Los elementos en común, en orden inverso.
      ii. Los elementos no comunes, en orden alfabético.

   El programa debería arrojar el siguiente resultado:
   ```
   listas(['b', 'a', 'c'], ['e', 'b', 'd', 'c'])
   ['c', 'b']
   ['a', 'e', 'd']
   ```

   e. Escribir un procedimiento *numeros_par_impar(entrada)* que, dada una lisa de números, eleve cada elemento impar en ella al cuadrado y los mueva a otra lista e imprima ambas. La lista de números la ingresa el usuario en forma de números separados por coma.
   Suponiendo que el usuario ingresa la siguiente lista:

```
1,2,3,4,5,6,7,8,9
```

Entonces, la salida del programa debería ser:

```
2,4,6,8
1,9,25,49,81
```

f. Un portal web requiere un formulario de alta de usuario donde se ingrese, mínimamente, un usuario y su correspondiente contraseña. Escriba, en Python, una función *contrasena_valida(contrasena)* que devuelva True en caso de superar las siguientes validaciones sobre la contraseña proporcionada por el usuario:

   i. Longitud entre 6 y 20 caracteres.
   ii. Debe contener al menos un número.
   iii. Debe contener al menos dos mayúsculas.
   iv. Debe contener al menos un carácter especial.
   v. No puede contener espacios.

La salida esperada es la siguiente:

```
abc.123 es válida:      False
Abc.123 es válida:      False
AbC.123 es válida:      True
AbC.1 23 es válida:     False
ÁbC.123 es válida:      False
```

Para la búsqueda de caracteres de cierto tipo (mayúsculas, acentos, espacios y otros) debe hacerse uso de la librería re:

- https://docs.python.org/es/3/library/re.html
- https://relopezbriega.github.io/blog/2015/07/19/expresiones-regulares-con-python/
- Para buscar caracteres especiales, puede utilizarse la siguiente expresión `[$&+,:;=?@#|<>.^*()%!-]`

2. Sección B: Introducción a la Programación Orientada a Objetos

A continuación, se presenta el código de un programa que, ante la edad ingresada por el usuario, este presenta el equivalente en días, meses y años. Se solicita al alumno que lo refactorice de manera tal que:

a. Se elimine la sentencia ***if / else*** de la función ***anio_bisiesto***.
b. Las múltiples sentencias ***if*** la función ***dia_mes*** utilicen la cláusula *in* en lugar de varias cláusulas ***or***.
c. Se agregue una sentencia que valide que la edad ingresada por el usuario es numérica.
d. Se agregue una función que encapsule el cálculo del equivalente de la edad en días y que tome como parámetros las variables ***hora_local***, ***anio_comienzo*** y ***anio_fin***.
e. Todas las funciones sean transportadas a un archivo auxiliar de funciones llamado funciones.py, y este sea importado desde el programa principal.

```
import time
from calendar import isleap

# calcular si es un año bisiesto
def anio_bisiesto(anio):
    if isleap(anio): return True
    else: return False

# calcular el numero de dias de cada mes
def calcular_dias_mes(mes, anio_bisiesto):
    if mes == 1 or mes == 3 or mes == 5 or mes == 7 or mes == 8 or mes ==
10 or mes == 12:
        return 31
    elif mes == 4 or mes == 6 or mes == 9 or mes == 11:
        return 30
    elif mes == 2 and anio_bisiesto == True:
        return 29
    elif mes == 2 and anio_bisiesto == False:
        return 28

# ingreso de datos del usuario
nombre = input("Ingrese su nombre: ")
edad = input("Ingrese su edad: ")
# seteo inicial de variables
hora_local = time.localtime(time.time())
anios = int(edad)
anio_comienzo = int(hora_local.tm_year) - anios
anio_fin = anio_comienzo + anios
meses = anios * 12 + hora_local.tm_mon
dias = 0

# calcular los dias
for a in range(anio_comienzo, anio_fin):
    if (anio_bisiesto(a)): dias = dias + 366
    else: dias = dias + 365

# agregar los días transcurridos en este año
for m in range(1, hora_local.tm_mon):
    dias = dias + calcular_dias_mes(m, anio_bisiesto(hora_local.tm_year))

# agregar los días transcurridos en este mes
dias = dias + hora_local.tm_mday

# imprimir la edad del usuario
print("La edad de %s es %d años o " % (nombre, anios), end="")
print("%d meses o %d días" % (meses, dias))
```

3. Sección C: Funciones matemáticas

Resolver cada ejercicio en un archivo diferente.

a. Escribir una función *suma(numero)* que resuelva la siguiente suma, asumiendo que *numero = 10*:

   1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10

   En el programa que invoque dicha función:

   i. El usuario debe poder ingresar el valor del parámetro *numero*.
   ii. Debe validarse que el dato ingresado por el usuario corresponda a un dígito, y no a otro tipo de dato como un carácter.
   iii. El cálculo debe realizarse utilizando algún tipo de bucle (ej: for, while).

   BONUS: Luego, codificar una función equivalente que utilice recursividad.

b. Escribir un programa que resuelva la secuencia de Fibonacci a pedido del usuario. Deberá codificar una función *fibonacci(numero)*, cuyo parámetro *numero* debe ser ingresado por el usuario y su tipo, al igual que en el ejercicio anterior, validado. La función debe encargarse de calcular la secuencia para dicho número. A continuación, una descripción matemática de la famosa secuencia:

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n > 1. \end{cases}$$

c. Tal como sucede con la lógica proposicional, en Python muchas veces las expresiones booleanas pueden ser simplificadas manteniendo el valor de verdad de la expresión. Así, por ejemplo, ***(a and b) or (b and a)*** es equivalente a *a* ***and*** *b*. A continuación, intente simplificar las siguientes expresiones y

escriba un procedimiento *procesar_sentencias(a, b, c)* que permita evaluar el valor de verdad de las expresiones ya simplificadas:

i. `(a or b) or (b and c)`

ii. `b and c or False`

iii. `a and b or c or (b and a)`

iv. `a == True or b == False`