UNER Facultad de Ciencias de la Administración

TRABAJO PRÁCTICO N° 1 – INTRODUCCIÓN A PYTHON

PROGRAMACIÓN II - 2025 – 2do cuatrimestre

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

Objetivos

- Repasar los conocimientos de Python adquiridos durante el cursado de Programación I.

Condiciones de Entrega

- Deberá realizarse una entrega grupal, con grupos de no más de 3 (tres) alumnos.
- Los ejercicios a entregar son los correspondientes a la sección “b”. La entrega deberá consistir de una carpeta comprimida que contenga 2 archivos: un archivo main.py y un archivo impresiones.py (para el ejercicio 8). Se deberá indicar con claridad la sección del archivo main que se corresponde con cada uno de los ejercicios del documento. La carpeta debe ser subida en la sección correspondiente del Campus.
- Se deberá respetar la fecha de entrega, la misma será informada en el campus.
- El Trabajo Práctico será calificado como **Aprobado** o **Desaprobado**.
- Las soluciones del grupo deberán ser de autoría propia. De encontrarse soluciones idénticas entre diferentes grupos, dichos trabajos prácticos serán calificados como **Desaprobado**, lo cual será comunicado en la devolución.
- Las entregas individuales serán calificadas como **Desaprobado**.

EJERCICIOS:

Sección a (para ver en clase):

1. Escribir un método recibir(). El mismo debe solicitar al usuario ingresar su nombre por consola e imprimir “Hola [nombre]”.
2. Escribir una función numero_no_contiene_digitos(numero, digitos_prohibidos), que retorne True si es que los dígitos que componen el número no se encuentran en una lista de dígitos prohibidos. Asumir que los valores pasados por parámetro son de un tipo válido.
3. Escribir un método invertir_palabras(entrada) que acepte una secuencia de palabras separadas por coma y la revierta. Suponiendo que la entrada provista al programa es la siguiente:

   el,día,está,soleado

   La salida esperada es:

   soleado,está,día,el
4. Escribir un método numeros_pares_elevados(entrada) que dado una lista de números eleve cada elemento par al cuadrado, los mueva a otra lista, e imprima dicha lista. Asumir que el parámetro es una lista de números positivos válida.
5. Escribir una función edad_valida(edad) que devuelva True si el valor pasado por parámetro es de tipo int, y si el mismo es mayor a 1 y menor a 120.
6. Reescribir la siguiente función eliminando la sentencia if/else y utilizando in en vez de or. Asumir que el parámetro es un valor válido.

```
def es_fin_de_semana(dia):
    if dia == 1 or dia == 7:
        return True
    else:
        return False
```

7. Dado el siguiente script, escribir una función calcular_hipotenusa(cateto1, cateto2) para aplicar el teorema de pitágoras, con el fin de mejorar la legibilidad del mismo. Incluir una validación para verificar que los catetos tienen longitud mayor a 0.

```
cateto1 = 5

cateto2 = 4

hipotenusa = (cateto1 ** 2 + cateto2 **2) ** 0.5

print(hipotenusa)

cateto1 = 3

cateto2 = 3

hipotenusa = (cateto1 ** 2 + cateto2 **2) ** 0.5

print(hipotenusa)

cateto1 = 7

cateto2 = 11.5

hipotenusa = (cateto1 ** 2 + cateto2 **2) ** 0.5

print(hipotenusa)
```

8. Extraer la función del ejercicio 7 a un archivo pitagoras.py, el cual debe ser importado para su utilización.
9. Escribir una función suma(entero_positivo) que imprima la suma de los números enteros por los que hay que pasar hasta llegar al número pasado por parámetro (incluido el mismo). Asumir que el número pasado por parámetro es un número entero positivo. Utilizar recursividad para desarrollar la solución.
10. Simplificar la siguiente expresión:

```
(a or b) and False
```

Sección b (para entregar):

1. Escribir un método realizar_calculo(). El mismo debe solicitar al usuario seleccionar por consola la operación matemática deseada (opciones: suma, resta, multiplicación, y división). Luego, debe solicitar que se ingresen 2 números. Por último, debe imprimir el resultado. Asumir que los valores ingresados son del tipo de datos correcto.
2. Escribir una función numero_en_orden_ascendente (numero) que retorne True si los dígitos del número están ordenados de menor a mayor, y False en caso contrario.

3. Escribir un método numeros_impares_juntos(entrada) que, dada una lista de números, concatene los números impares en un string, separados por coma.
4. Escribir un método lista_elementos_en_comun(lista1, lista2) que imprima los elementos en común que tienen las listas sin repetirlos.
5. Escribir una función clave_valida(clave) que devuelva True en caso de superar las siguientes validaciones sobre la clave proporcionada por el usuario:
    a. Longitud entre 6 y 20 caracteres.
    b. Debe contener al menos un número.
    c. No puede contener espacios.
6. Reescribir la siguiente función eliminando la sentencia if/else:

```
def persona_mayor_de_edad(edad):
    if edad >= 18:
        return True
    else:
        return False
```

7. Dado el siguiente script, escriba una función declarar_comida_favorita(nombre_persona, nombre_comida) con el fin de mejorar la legibilidad del mismo.

```
nombre = "Pablo"
comida_favorita = "pollo frito"
print("La comida favorita de " + nombre + " se llama: " + comida_favorita)
nombre = "Pedro"
comida_favorita = "canelones"
print("La comida favorita de " + nombre + " se llama: " + comida_favorita)
nombre = "Juan"
comida_favorita = "pizza"
print("La comida favorita de " + nombre + " se llama: " + comida_favorita)
```

8. Extraer la función del ejercicio 7 a un archivo impresiones.py, el cual debe ser importado para su utilización.

9. Escribir una función cuenta_regresiva(entero_positivo) que imprima números enteros empezando por el valor pasado por parámetro y llegando hasta 0. Asumir que el número pasado por parámetro es un número entero positivo. Utilizar recursividad para desarrollar la solución.

10. Simplificar la siguiente expresión:

(a and b) or True