UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA
MATERIA: PROGRAMACIÓN II
2do Parcial
Teórico y Práctico

Alumno: Lastra Julian Marcos **Fecha:** 19/11/22 **Tema 1**
Comisión – Localización - Turno: 3 A - Lomas - Noche
Práctica: **Teoría:** **Nota:**

Temas para evaluar: Paradigma: Funcional, integración con POO y paradigma estructurado. Archivos, excepciones, librerías, introducción al machine learning

Objetivos:

Comprender cómo se aplican las técnicas del paradigma funcional en lenguaje python
Comprender cómo se gestionan los archivos, módulos y paquetes
Comprender cómo realizar un buen tratamiento de excepciones
Comprender y utilizar librerías y los principales algoritmos de deep learning

Modalidad: Parcial domiciliario

Requisitos para aprobar: Para que el parcial esté aprobado el alumno deberá tener correctamente desarrolladas el 60% de la teoría y resuelto el ejercicio práctico.

Tiempo:

Recomendaciones:

a) Lea todo el parcial antes de comenzar a responder.
b) Desarrolle una redacción clara y precisa contestando lo que la pregunta requiere.
c) Observe la ortografía ya que la misma es parte del parcial.
d) Si considera que no comprende alguna consigna antes de comenzar consulte a su profesor.

Notas: Las preguntas en las que se seleccionen opciones se deberá optar solo por una de las posibilidades. La indicación se efectuará con una X sobre su lateral izquierdo, será considerada nula si presenta tachaduras o enmiendas.
Las preguntas que solicitan justificación serán consideradas válidas si poseen la misma correctamente.
Las preguntas de múltiples posibilidades y verdadero / falso restan 0.50 puntos en caso de estar mal contestadas. En las preguntas verdadero / falso se debe tachar la opción incorrecta.

(*) la cifra entre paréntesis en cada pregunta es la cantidad de puntos sobre 100.

1. Indique tres diferencias sustanciales al momento de diseñar un programa utilizando el paradigma funcional y el POO.

La diferencia más clara a la hora de diseñar un programa utilizando el paradigma funcional vs POO, está en que el primero está compuesto únicamente por funciones, contra POO, que se basa en toda la teoría de objetos para poder realizar métodos.

En el paradigma funcional nos basamos en funciones de orden superior mayormente para realizar el programa, en cambio en POO, lo que hacemos es relacionar las diferentes clases y sus métodos para pasarnos información o procesarla.

Por último, en el paradigma funcional tenemos que ser más cuidadosos a la hora de escribir código, dado que nos permite utilizar variables "inmutables", a diferencia de POO que es fuertemente tipado, y es más fácil encontrar errores con respecto a tipos de variables.

2. Ejemplifique el uso de una función de orden superior (10)

```
#Punto 2. Funcion de orden superior
def alquilarAuto(auto):
    def alquilar_bwm():
        print("Alquilo un BMW")
    def alquilar_audi():
        print("Alquilo un Audi")
    alquilar_func = {
        "bmw" : alquilar_bwm,
        "audi" : alquilar_audi
    }
    return alquilar_func[auto]
alquilarAuto("bmw")()
```

3. ¿Qué diferencias hay entre map, filter y reduce? (10)

La función MAP, lo que hace en simples palabras es comparar elementos y arrojar un resultado en caso de haber coincidencia.
Por ejemplo esta función es muy útil cuando queremos "mapear" un JSON a una Clase cualquiera, utilizamos el wildcard "**" para que haga automáticamente el "mapeo".

La función filter, lo que hace es verificar que los elementos de una secuencia cumplan con una condición determinada.
Por ejemplo podemos hacer una lista en donde solo los números "filtrados" sean mayores a 10.

La función reduce lo que hace es aplicar una función a 2 elementos de una secuencia hasta que queda un solo resultado.
Por ejemplo agarra una lista de números y los multiplica a todos hasta obtener el resultado.

Como vemos, cada una cumple una función completamente diferente sobre una secuencia.

4. ¿Qué es una función lambda? Ejemplifique con una porción de código. (10)

La función lambda sirve para crear funciones anónimas en línea. Dado que son anónimas (no tienen nombre), no pueden ser referenciadas, únicamente van a ser utilizadas al momento de "castearlas". Estas funciones se componen mediante un operador al que llamamos lambda.

Por ejemplo acá tenemos una función que lo que hace es agarrar el número de multiplicador(), y luego lo multiplica a cada uno de los números que están dentro de cantidadMults (). En este caso tenemos 222 que se va a multiplicar individualmente por 3.

```
#Punto 4. Función Lambda
def multiplicador(n):
  return lambda a : a * n
cantidadMults = multiplicador(3)
print(cantidadMults(222))
#>>666
```

5. ¿Qué es una comprensión de listas? Ejemplifique (10)

Comprensión de listas es una característica que fue tomada del lenguaje de programación funcional Haskell, que consiste en una construcción que permite crear listas a partir de otras listas. Cada una de estas construcciones consta de una expresión que determina cómo modificar el elemento de la lista original, seguida de una o varias clausulas for y opcionalmente una o varias clausulas if.
A diferencia de reduce, comprensión de listas está incluida en Python desde la versión 2 y no es necesario llamar a algún modulo adicional.

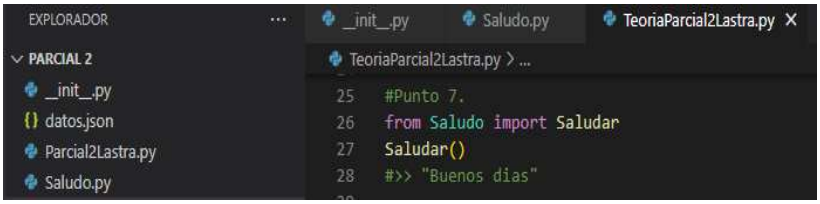
```
#Punto 5. Comprension de listas. En este caso lo que hacemos es quitar elementos de una lista que sean "audi".
autos = ["bmw", "audi", "ford", "Lamorbigini", "Paganni"]
listaSinAudis = [x for x in autos if x != "audi"]
```

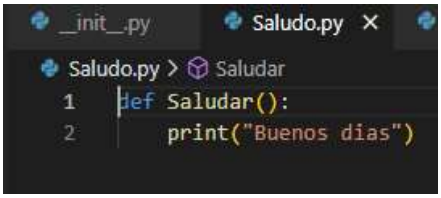
6. ¿Cuál es la diferencia entre un generador y un decorador? (10)

Las expresiones generadoras funcionan de forma muy similar a la comprensión de listas. De hecho, su sintaxis es exactamente igual, a excepción de que se utilizan paréntesis en lugar de corchetes. Sin embargo, las expresiones generadoras se diferencian de la comprensión de listas en que no se devuelve una lista, sino un generador.

En cambio el decorador no es más que una función que recibe una función como parámetro y devuelve otra función como resultado. Por ejemplo, podríamos querer añadir la funcionalidad de que se imprimiera el nombre de la función llamada por motivos de depuración.

7. Ejemplifique el uso de un paquete y un módulo en python (10)





8. ¿Cómo aplica try y except en el uso de archivos? (10)

Para el uso de archivos lo primero que implementaría sería como última opción a una "Exception" general. Para capturar cualquier tipo de error que no pudiera haber previsto.
Una de las primero que aplicaría seria la excepción "OSError", dado que la misma nos avisa si el archivo no puede ser abierto, por ejemplo.
Otra de las que usaría es la excepción "FileNotFoundError", que se refiere a cuando no se encuentra el archivo que estamos buscando.

Codigo ejemplo:

```
try:
    #Sentencia de codigo
except FileNotFoundError as e_fnf:
    print("no se encontro el archivo") print(e_fnf)
except OSError as e_os:
    print("no se pudo abrir el archivo") prnt(e_os)
except Exception as e_gen:
   print("Ocurrio una excepcion al abrir el archivo") print(e_gen)
```

9. Cómo sería una expresión regular para validar un email con solo números antes del @ y que siempre sea del dominio uai.edu.ar. Ejemplo: 186333@uai.edu.ar (10)

La expresión regular seria "[0-9]+@uai.edu.ar\Z". El [0-9] hace que solo números puedan ir delante de @. El + es para que tome más de 1 ocurrencia con respecto a los números. el @uai.edu.ar\Z, hace que la sentencia tenga que finalizar con ese parámetro especifico. Para realizar la comprobación podemos utilizar el código:

```
import re
txt = "51234@uai.edu.ar"
x = re.findall("[0-9]+@uai.edu.ar\Z", txt)
print(x)
```

Si concuerda con lo que buscábamos nos va a devolver txt, en caso de que no concuerde nos arrojara "[]".

10. ¿Cuál es la diferencia entre machine learning y Deep learning? (10)

Principalmente encontramos 4 grandes diferencias entre machine learning y deep learning.
1. La intervencion humana. Mientras que machine learning requiere intervension humana constante para intervenir y obtener los resultados, el deep learning es más complejo de "preparar" pero una vez que eso sucede, se rquiere intervencion humana minima.

2. Con respecto al Hardware, dado que los programas de machine learning tienden a ser menos complejos que los algoritmos de deep learning, se requiere menos hardware para poder correr uno o el otro.

3. Con respecto al tiempo los sistemas basados en machine learning, pueden "setupearse" más rápido que los de deep learning, aunque estos últimos generan resultados casi instantáneamente comparados con los de ML.

4. El modo en el que atacan los problemas. Machine learning utiliza información estructurada y a su vez utiliza algoritmos tradicionales como la regresión lineal. En cambio deep learning, utiliza redes neuronales y está preparado para data que no está estructurada.

UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA
MATERIA: PROGRAMACIÓN II
2do Parcial
Teórico y Práctico

PRÁCTICA

Genere un programa que permita realizar la estimación de recorridos de una flota de camiones a lo largo de todo el país.

1) Cada clase Camion posee una patente, litros_disponibles, ciudad_actual, km_litro, vel_maxima y una clase Chofer (Con datos a definir por usted). Además, cada camión realiza recorridos que pasan por diversas ciudades entregando la mercadería de la empresa. (20)
2) El programa debe permitir elegir un camión a partir de su patente y cargar un recorrido entre distintas ciudades (Se pueden prefijar cuatro o cinco ciudades por defecto con las respectivas distancias entre ellas). (10)
3) Una vez escogido el camión y cargadas las ciudades destino, el programa devolverá a partir de un método llamado estimar_viaje(Camion, Ciudad[]):Estimacion los km por recorrer totales, el tiempo total estimado y si el combustible del camión alcanza para realizar todo el trayecto. Para hacer el cálculo utilizar la vel_maxima como velocidad promedio entre cada tramo y en cada ciudad sumar 60 minutos de parada. (10)
4) Cada vez que se solicite el método estimar_viaje() se devolverá un objeto Estimacion con todos los datos solicitados, antes de retornar el objeto, se deberá persistir en un archivo toda la información calculada junto a la patente del camión. (20)
5) Permitir buscar todas las estimaciones realizadas para una patente X dentro del archivo guardado en el punto 4, mostrar por consola qué viajes superaron los 1000 km de distancia total y cuáles no. (20)
6) Genere código python para probar la solución propuesta y la prueba del mensaje (20)

Dentro del código entregado deberá haber utilizado al menos dos de estas características funcionales: Map, filter, reduce, expresiones lambda, generadores, decoradores o comprensión de listas. Hacer un buen manejo de las excepciones. Cualquier archivo de configuración inicial que desee agregar será de suma importancia.

Lastra, Julián Marcos
Firma del Alumno