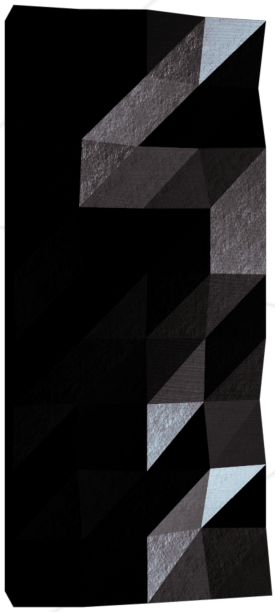

# Entrenamiento de Piscine Python para la ciencia de datos - 4

## Diseño orientado a datos

*Resumen: Hoy verás algo de diseño de estructura.*

*Versión: 1.00*

# Contenido

# Capítulo I

## Reglas generales

- Debes renderizar tus módulos desde una computadora en el clúster usando una máquina virtual:
    - Puede elegir el sistema operativo que utilizará para su máquina virtual
    - Su máquina virtual debe contar con todo el software necesario para realizar su proyecto. Este software debe estar configurado e instalado.
- O puedes utilizar directamente la computadora en caso de que tengas las herramientas disponibles.
    - Asegúrate de tener el espacio en tu sesión para instalar lo que necesitas para todos los módulos (usa goinfre si tu campus lo tiene)
    - Debes tener todo instalado antes de las evaluaciones.
- Sus funciones no deberían cerrarse inesperadamente (error de segmentación, error de bus, doble liberación, etc.) salvo por comportamientos indefinidos. Si esto sucede, su proyecto se considerará no funcional y recibirá una0Durante la evaluación.
- Le animamos a crear programas de prueba para su proyecto, incluso si este trabajo no es suficiente. **No será necesario enviarlo y no será calificado.**. Te dará la oportunidad de evaluar fácilmente tu trabajo y el de tus compañeros. Estas pruebas te resultarán especialmente útiles durante tu defensa. De hecho, durante la defensa, eres libre de utilizar tus propias pruebas y/o las pruebas de los compañeros a los que estás evaluando.
- Envía tu trabajo al repositorio git que se te haya asignado. Solo se calificará el trabajo que se encuentre en el repositorio git. Si se le asigna a Deepthought la tarea de calificar tu trabajo, se hará después de las evaluaciones de tus pares. Si ocurre un error en alguna sección de tu trabajo durante la calificación de Deepthought, la evaluación se detendrá.
- Debes utilizar la versión Python 3.10
- Las importaciones de bibliotecas deben ser explícitas, por ejemplo, debe "importar numpy como np". No se permite importar "from pandas import *" y obtendrá 0 en el ejercicio.
- No hay ninguna variable global.
- ¡Por Odín, por Thor! ¡Usa tu cerebro!

# Capítulo II

# Instrucciones específicas del día

Una queja común de los científicos de datos es que escriben código basura (por cierto, solo con fines educativos puedes encontrar muchos ejemplos de código basura de Python).aquí, proporcionado estrictamente con fines educativos). ¿Por qué? Porque el científico de datos promedio utiliza muchas técnicas ineficientes y variables codificadas de forma rígida y descuida la programación orientada a objetos. No sea como ellos.

- No hay código en el ámbito global. ¡Utilice funciones!
- Cada programa debe tener su main y no ser un simple script:

```
definición principal():
    # sus pruebas y su manejo de errores

si __nombre__ == "__principal__":
    principal()
```

- Cualquier excepción no detectada invalidará los ejercicios, incluso en el caso de un error que se le solicitó probar.
- Puedes utilizar cualquier función incorporada si no está prohibida en el ejercicio.
- Todas sus funciones, clases y métodos deben tener una documentación (__doc__)
- Su código debe estar en la norma
    - pip instala flake8
    - alias norminette=flake8

# Capítulo III

# Ejercicio 00

|  | Ejercicio 00 |
|---|---|
| Ejercicio 00: Calcular mis estadísticas | |
| Directorio de entregas:*ex*00*/* | |
| Archivos a entregar:estadísticas.py | |
| Funciones permitidas:Ninguno | |

Debes tomar en *args una cantidad de número desconocido y hacer la Media, Mediana, Cuartil (25% y 75%), Desviación Estándar y Varianza de acuerdo a la pregunta de **kwargs.

Hay que gestionar los errores.
El prototipo de función es:

```
def ft_estadísticas(*argumentos:Cualquier,**kurangas:Cualquier)->Ninguno:
  # tu código aquí
```

Su tester.py:

```
Desde las estadísticas, importe ft_statistics

ft_estadísticas(1,42,360,11,64, toto="significar", tutú="mediana", tía="cuartilla") imprimir("-----"
)
ft_estadísticas(5,75,450,18,597,27474,48575, Hola="estándar", mundo="var") imprimir(
"-----")
ft_estadísticas(5,75,450,18,597,27474,48575, ejfhhe="jejeje", ejdjdejn="kdekem") imprimir("-----")

ft_estadísticas(toto="significar", tutú="mediana", tía="cuartilla")
```

Resultado esperado:

```
> probador de python.py
media: 95,6
mediana: 42
cuartil: [11.0, 64.0]
- - - - -
estándar: 17982.70124086944
var: 323377543.9183673
- - - - -
- - - - -
ERROR
ERROR
ERROR
$>
```

# Capítulo IV

# Ejercicio 01

|  | Ejercicio 01 |
|---|---|
| Ejercicio 01: Exterior_interior | |
| Directorio de entregas:*ex*01/ | |
| Archivos a entregar:dentro_fuera.py | |
| Funciones permitidas:Ninguno | |

Escribe una función que devuelva el cuadrado del argumento, una función que devuelva la exponenciación del argumento por sí misma y una función que tome como argumento un número y una función, devuelva un objeto que al ser llamado devuelva el resultado del cálculo de los argumentos.

El prototipo de funciones es:

```
def cuadrado(x:entero|flotar)->entero|flotar:
  # tu código aquí

definición potencia(x):entero|flotar)->entero|flotar:
  # tu código aquí

def exterior(x:entero|flotar, función)->objeto:
                contar= 0
        definición interna()->flotar:
          # tu código aquí
```

Su tester.py:

```
desde in_out importar exterior
desde in_out importar cuadrado
desde in_out importar pow

mi_contador=exterior(3, cuadrado)
print(mi_contador())
imprimir(mi_contador())
imprimir(mi_contador())
imprimir("---")
otro_contador=exterior(1.5, pow)
imprimir(otro_contador())
imprimir(otro_contador())
imprimir(otro_contador())
```

Resultado esperado:

```
> probador de python.py
9
81
6561
- - -
1.8371173070873836
3.056683336818703
30.42684786675409
$>
```

Le recordamos que el uso de global está prohibido

# Capítulo V

# Ejercicio 02

| | |
|---|---|
|  | Ejercicio 02 |
| Ejercicio 02: mi primera decoración | |
| Directorio de entregas:*ex*02/ | |
| Archivos a entregar:llamadaLimit.py | |
| Funciones permitidas:Ninguno | |

Escriba una función que tome como argumento un límite de llamada de otra función y bloquee su ejecución por encima de un límite.

El prototipo de funciones es:

```
def callLimit(límite:entero):
    contar= 0
    def callLimiter(función):
        def función_limite(*argumentos:Cualquier,**kwds:Cualquier):
          # tu código aquí
```

Su tester.py:

```
desde callLimit importar callLimit

límite de llamadas(3)
definición f():
    imprimir ("F()")

límite de llamadas(1)
definición g():
    imprimir ("gramo()")

parayo en rango(3):
    F()
    gramo()
```

Resultado esperado:

```
> probador de python.py
F()
gramo()
F()
Error: <función g en 0x7fabdc243ee0> llama demasiadas veces a f()

Error: <función g en 0x7fabdc243ee0> se llama demasiadas veces $>
```

Envoltorios

# Capítulo VI

# Ejercicio 03

|  | Ejercicio 03 |
|---|---|
| Ejercicio 03: clase de datos | |
| Directorio de entregas:*ex*03/ | |
| Archivos a entregar:nuevo_estudiante.py | |
| Funciones permitidas:clases de datos, aleatorio, cadena | |

Escriba una clase de datos que tome como argumentos un nombre y un apodo, establezca active en True, cree el inicio de sesión del estudiante y genere una identificación aleatoria con la función generate_id. No debe usar __str__ , __repr__ en su clase.

El prototipo de función y clase es:

```
importar aleatorio
cadena de importación
desde dataclasses importar dataclass, campo

def generar_id()->cadena:
        devolver"".join(opciones.aleatorias(cadena.ascii_minúsculas, k= 15))

clase de datos
Clase Estudiante:
  # tu código aquí
```

Su tester.py:

```
desde new_student importar Estudiante

alumno=Estudiante(nombre="Eduardo", apellido="águila")
imprimir(estudiante)
```

Resultado esperado: (id es aleatorio)

```
> probador de python.py
Estudiante(nombre='Edward', apellido='agle', activo=True, login='Eagle', id='trannxhndgtolvh') $>
```

El nombre de usuario y la identificación no deben ser inicializables y deben devolver un error.

Su tester.py:

```
desde new_student importar Estudiante

alumno=Estudiante(nombre="Eduardo", apellido="águila", identificación="Toto")
imprimir(estudiante)
```

Resultado esperado:

```
> probador de python.py
. . .
TypeError: Student.__init__() obtuvo un argumento de palabra clave inesperado 'id'
$>
```

# Capítulo VII

## Presentación y evaluación por pares

Entregue su tarea en suGitRepositorio como de costumbre. Solo el trabajo dentro de su repositorio será evaluado durante la defensa. No dude en volver a verificar los nombres de sus carpetas y archivos para asegurarse de que sean correctos.

El proceso de evaluación se realizará en el computador del grupo evaluado.