



Entrenamiento Piscine Python para datascience - 1

Formación

Resumen: Hoy descubrirás las matrices, sus manipulaciones y el trabajo con imágenes.

Versión: 1.1

Contenido

I	Reglas generales	2
II	Instrucciones específicas del día	3
III	Ejercicio 00	4
IV	Ejercicio 01	5
V	Ejercicio 02	6
VI	Ejercicio 03	7
VII	Ejercicio 04	9
VIII	Ejercicio 05	11
IX	Presentación y evaluación por pares	14

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 una 0 Durante 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


- 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 tus funciones 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: Dame mi IMC	
Directorio de entregas: ex00/	
Archivos a entregar: dar_bmi.py	
Funciones permitidas: numpy o cualquier biblioteca de manipulación de tablas	

Su función, `give_bmi`, toma 2 listas de números enteros o flotantes en la entrada y devuelve una lista de valores de IMC.

Su función, `apply_limit`, acepta una lista de números enteros o flotantes y un número entero que representa un límite como parámetros. Devuelve una lista de valores booleanos (verdadero si el valor es superior al límite).

Debes manejar casos de error si las listas no tienen el mismo tamaño, no son int o float...

El prototipo de funciones es:

```
def give_bmi(altura: lista[entero | flotar], peso: lista[entero | flotar]) -> lista[entero | flotar]:  
    # tu código aquí  
  
def aplicar_limite(imc: lista[entero | flotar], límite: entero) -> lista[Booleano]:  
    # tu código aquí
```

Su `tester.py`:


```
desde give_bmi importar give_bmi, aplicar_limit  
  
altura=[2.71,1.15] peso=[  
165.3,38.4]  
  
IMC=give_bmi(altura, peso)  
print(imc, tipo(imc))  
imprimir(aplicar_limite(imc,26))
```

Resultado esperado:

```
> probador de python.py  
[22.507863455018317, 29.0359168241966] <clase 'lista'> [Falso,  
Verdadero]  
$>
```

Capítulo IV

Ejercicio 01

	Ejercicio 01
Ejercicio 01: Matriz 2D	
Directorio de entregas: ex01/	
Archivos a entregar: matriz2D.py	
Funciones permitidas: numpy o cualquier biblioteca de manipulación de tablas	

Escriba una función que tome como parámetros una matriz 2D, imprima su forma y devuelva una versión truncada de la matriz en función de los argumentos de inicio y final proporcionados. Debes utilizar el método de corte. Debes manejar casos de error si las listas no tienen el mismo tamaño, no son una lista...

El prototipo de función es:

```
def slice_me(familia: lista, inicio: entero, fin: entero) -> lista:  
    # tu código aquí
```

Su tester.py:


```
desde array2D importar slice_me  
  
familia=[[1,80,78.4],  
         [2.15,102.7], [  
         2.10,98,5], [1,88,  
         75.2]]  
  
imprimir(slice_me(familia,0,2))  
imprimir(slice_me(familia,1,-2))
```

Resultado esperado:

```
$> python test_array2D.py Mi  
forma es: (4, 2)  
Mi nueva forma es: (2, 2) [[1.8,  
78.4], [2.15, 102.7]] Mi forma es:  
(4, 2)  
Mi nueva forma es: (1, 2)  
[[2.15, 102.7]]  
$>
```

Capítulo V

Ejercicio 02

	Ejercicio 02
Ejercicio 02: cargar mi imagen	
Directorio de entregas: ex02/	
Archivos a entregar: cargar_imagen.py	
Funciones permitidas: Todas las bibliotecas para cargar imágenes y manipulación de tablas.	

Necesita escribir una función que cargue una imagen, imprima su formato y su contenido de píxeles en formato RGB.

Debes manejar, al menos, el formato JPG y JPEG. Debes manejar cualquier error con un mensaje de error claro.

Así es como debería ser el prototipo:

```
def ft_load(ruta:cadena)->formación:(puede devolver al formato deseado)
# tu código aquí
```

Su tester.py:


```
desde load_image importar ft_load
imprimir(ft_load("paisaje.jpg"))
```

Resultado esperado:

```
> probador de python.py
La forma de la imagen es: (257, 450, 3) [[[19
42 83]
 [23 42 84]
 [28 43 84]
 ...
 [ 0  0  0]
 [ 1  1  1]
 [ 1  1  1]]]
$>
```

Capítulo VI

Ejercicio 03

	Ejercicio 03
Ejercicio 03: acércate a mí	
Directorio de entregas: <i>ex03/</i>	
Archivos a entregar: <i>cargar_imagen.py</i> , <i>zoom.py</i>	
Funciones permitidas: Todas las bibliotecas para cargar, manipular, mostrar imágenes y manipulación de tablas.	

Cree un programa que debería cargar la imagen "animal.jpeg", imprimir alguna información sobre ella y mostrarla después de hacer "zoom".

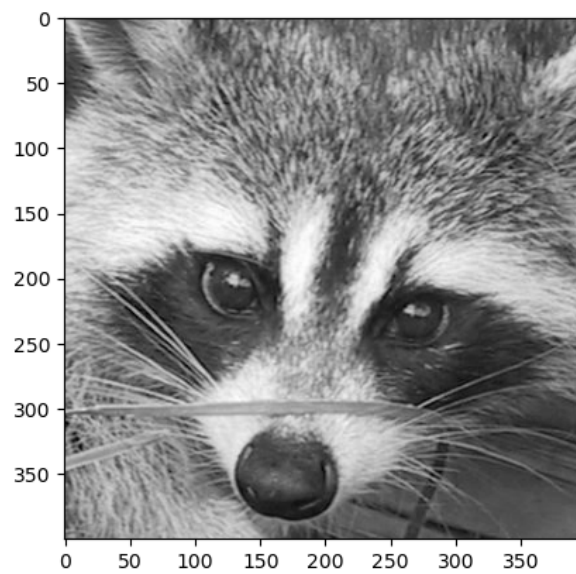
- El tamaño en píxeles en los ejes X e Y
- El número de canal
- El contenido de píxeles de la imagen.
- Muestra la escala en los ejes x e y en la imagen

Si algo sale mal, el programa no debe detenerse abruptamente y manejar cualquier error con un mensaje claro.

Resultado esperado:

```
> zoom de python.py
La forma de la imagen es: (768, 1024, 3)
[[[120 111 132]
  [139 130 151]
  [155 146 167]
  ...
  [120 156 94]
  [119 154 90]
  [118 153 89]]]
Nueva forma después del corte: (400, 400, 1) o (400, 400)
[[[167]
  [180]
  [194]
  ...
  [102]
  [104]
  [103]]]
$>
```



Resultado esperado:



La matriz después del corte y el área de zoom pueden ser diferentes.

Capítulo VII

Ejercicio 04

	Ejercicio 04
Ejercicio 04: rotame	
Directorio de entregas: <i>ex04/</i>	
Archivos a entregar: <i>cargar_imagen.py</i> , <i>rotar.py</i>	
Funciones permitidas: Todas las bibliotecas para cargar, manipular, mostrar imágenes y manipulación de tablas.	

Cree un programa que cargue la imagen "animal.jpeg", corte una parte cuadrada de ella y transpóngala para producir la imagen que se muestra a continuación. Debería mostrarla, imprimir la nueva forma y los datos de la imagen después de la transposición.

Resultado esperado:

```
$> python rotar.py
La forma de la imagen es: (400, 400, 1) o (400, 400) [[[167]
[180]
[194]
...
[102]
[104]
[103]]]
Nueva forma después de la transposición: (400,
400) [[167 180 194 ... 64 50 72]
...
[115 116 119 ... 102 104 103]] $>
```

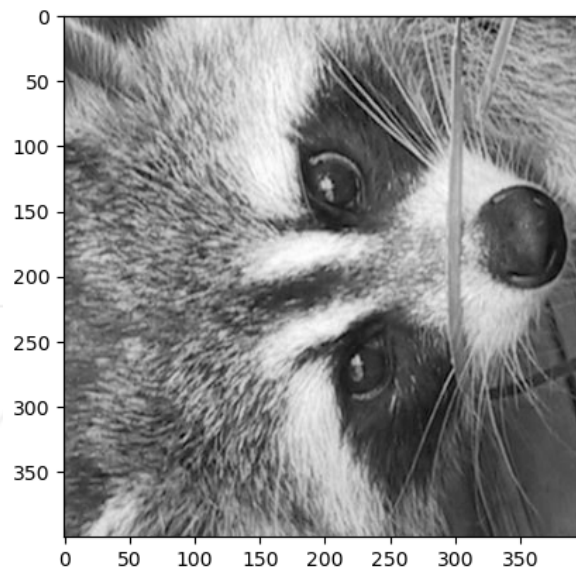


Su matriz después de la transposición puede ser diferente.
Puedes buscar el método de transposición, podría ayudarte...




Debes realizar la transposición tú mismo, no se permite ninguna biblioteca para la transposición.

Resultado esperado:



Capítulo VIII

Ejercicio 05

	Ejercicio 05
Ejercicio 05: Mejora mi imagen	
Directorio de entregas: ex05/	
Archivos a entregar: cargar_imagen.py, pimp_image.py	
Funciones permitidas: Todas las bibliotecas para cargar, manipular, mostrar imágenes y manipulación de tablas.	

Necesita desarrollar 5 funciones capaces de aplicar una variedad de filtros de color a las imágenes, manteniendo la misma forma de la imagen.

Así es como deberían ser prototipados:

```
def ft_invert(matriz)->formación:
    # tu código aquí

def ft_red(matriz)->formación:
    # tu código aquí

def ft_green(matriz)->formación:
    # tu código aquí

def ft_blue(matriz)->formación:
    # tu código aquí

def ft_grey(matriz)->formación:
    # tu código aquí
```

Tienes algunos operadores de restricción para cada función: (solo puedes usar los que se dan, no tienes que usarlos todos)

- invertir: =, +, -, *
- rojo: =, *
- verde: =, -
- azul: =
- gris: =, /

Su tester.py:

```
desde load_image importar ft_load
desde pimg_image importar ft_invert
...

formación=carga_pie("paisaje.jpg")

ft_invert(matriz)
ft_red(matriz)
ft_green(matriz)
ft_blue(matriz)
ft_grey(matriz)

imprimir(ft_invert.__doc__)
```

Resultado esperado: (las cadenas de documentación pueden ser diferentes)

```
> probador de python.py
La forma de la imagen es: (257, 450, 3) [[[19
42 83]
 [23 42 84]
 [28 43 84]
 ...
 [ 0  0  0]
 [ 1  1  1]
 [ 1  1  1]]]
...
Invierte el color de la imagen recibida. $>
```

Resultado esperado: (debes mostrar las imágenes transformadas)



Figura VIII.1: Original

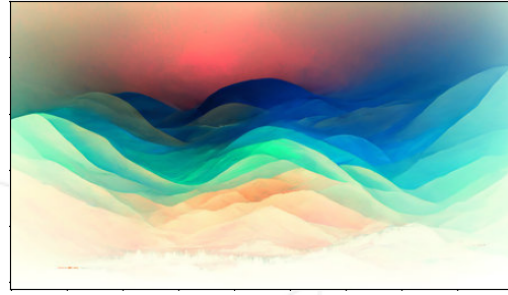


Figura VIII.2: Invertir



Figura VIII.3: Rojo

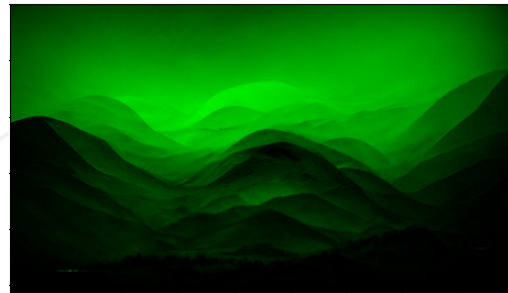


Figura VIII.4: Verde

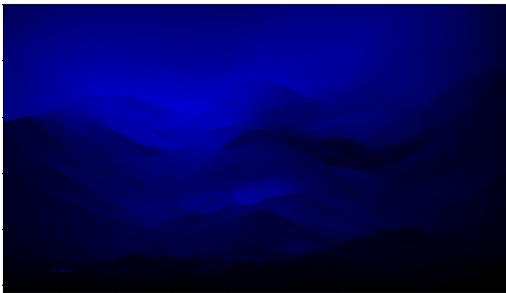


Figura VIII.5: Azul



Figura VIII.6: Gris

Capítulo IX

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.