Introducción

Antes de empezar con las lecciones es muy importante que cree un directorio (carpeta) en su repositorio, el que venimos trabajando el clase, en este caso como son las primeras lecciones primero deberá crear un directorio llamado **lecciones** , para crearlo deberá usar la terminal (consola) y verificar con el comando **pwd** que esté en la ruta inicial de su proyecto y ejecutar el comando **mkdir lecciones** , luego de crear el directorio de lecciones procedemos a ingresar en él con el comando **cd lecciones** , una vez que estamos en directorio “lecciones” creamos nuestro directorio para resolver todas estas lecciones de esta unidad, deberá crearlo con el comando **mkdir lecciones_01** , donde creará todos los programas (los archivos con extensión .py) para realizar las lecciones.

Resumen de pasos:
cd (para ir a la ruta inicial de mi proyecto).
pwd (para verificar dónde estoy).
mkdir lecciones (para crear directorio principal de todas las lecciones de este curso).
cd lecciones (para acceder al directorio).
mkdir lecciones_01 (para crear el directorio para estas lecciones).
cd lecciones_01 (para acceder al directorio).

Voz interior

ESCRIBIR EN MAYÚSCULAS ES COMO GRITAR.
Lo mejor es utilizar a veces la “voz interior” y escribir enteramente en minúsculas.
En un archivo llamado **voz_interior.py** , implemente un programa en Python que solicite al usuario una entrada y la muestre en minúsculas. La puntuación y los espacios en blanco deben mostrarse sin cambios. Puede solicitar la entrada explícitamente, aunque no es obligatorio, por ejemplo, pasando un **str** propio como argumento a **input**.

Pistas

Recuerde que **input** devuelve un **str** , según
docs.python.org/3/library/functions.html#input
Recuerde que a str viene con bastantes métodos, según
https://docs.python.org/3/library/stdtypes.html#string-methods

Demostración

```
$ python indoor.py
HELLO, WORLD
hello, world
$
```

Antes de empezar

Ejecuta:

mkdir voz_interior

para crear una carpeta llamada voz_interior en tu espacio de código.

Después ejecuta:

cd voz_interior

para cambiar directorios a esa carpeta. Ahora deberías ver tu prompt de terminal como voz_interior/ $.

Ahora puede ejecutar:
code voz_interior.py
para crear un archivo llamado voz_interior.py donde escribirás tu programa.

Cómo Probar

Aquí te mostramos cómo probar tu código manualmente:

Ejecuta tu programa con python voz_interior.py. Escribe HOLA y presiona Enter. Tu programa debería mostrar:

hola

Ejecuta tu programa con python voz_interior.py. Escribe CURSO PYTHON y presiona Enter. Tu programa debería mostrar:

curso python

Ejecuta tu programa con python voz_interior.py. Escribe 100 y presiona Enter. Tu programa debería mostrar:

100

Velocidad de reproducción

Algunas personas tienen la costumbre de dar conferencias hablando bastante rápido, y sería bueno ralentizarse, al estilo de la velocidad de reproducción de 0.75 de YouTube, o incluso haciendo que hagan pausas entre palabras.
En un archivo llamado reproduccion.py, implementa un programa en Python que solicite al usuario una entrada y luego muestre esa misma entrada, reemplazando cada espacio con ... (es decir, tres puntos).

Pistas

- Recuerda que input devuelve un str, según docs.python.org/3/library/functions.html#input.

- Recuerda que un str viene con bastantes métodos, según docs.python.org/3/library/stdtypes.html#string-methods.

Demostración

```
$ python reproduccion.py
Esto es curso de Python
 Esto...es...curso…de…Python.
$
```

Antes de Comenzar

Ejecuta:

mkdir reproduccion

para crear una carpeta llamada reproduccion en tu espacio de código.

Después ejecuta:

cd reproduccion

para cambiar directorios a esa carpeta. Ahora deberías ver tu prompt de terminal como reproduccion/ $.

Ahora puedes ejecutar:

code reproduccion.py

para crear un archivo llamado reproduccion.py donde escribirás tu programa.

Cómo Probar

Aquí te mostramos cómo probar tu código manualmente:

Ejecuta tu programa con python reproduccion.py. Escribe Esto es curso de Python y presiona Enter. Tu programa debería mostrar:

```
Esto...es...curso…de…Python
```

Ejecuta tu programa con python reproducción.py. Escribe Esto es nuestra semana de funciones y presiona Enter. Tu programa debería mostrar:

```
Esta...es...nuestra...semana...de...funciones
```

Ejecuta tu programa con python reproduccion.py. Escribe vamos a implementar una funcion llamada hola y presiona Enter. Tu programa debería mostrar:

```
vamos...a...implementar...una…funcion…llamada…hola
```

Haciendo caras

Antes de que existieran los emoji, existían los emoticones, donde texto como `:)` era una cara feliz y texto como `:(` era una cara triste. ¡Hoy en día, los programas tienden a convertir emoticones a emoji automáticamente!

En un archivo llamado `caras.py`, implementa una función llamada `convertir` que acepte un `str` como entrada y devuelva esa misma entrada con cualquier `:)` convertido a 🙂 (también conocido como cara ligeramente sonriente) y cualquier `:(` convertido a 🙁 (también conocido como cara ligeramente fruncida). Todo el texto demás debe devolverse sin cambios.

Luego, en ese mismo archivo, implementa una función llamada `main` que solicite al usuario una entrada, llame a `convertir` con esa entrada, e imprima el resultado. Eres bienvenido, pero no requerido, a solicitar explícitamente al usuario, como pasando un `str` propio como argumento a `input`. Asegúrate de llamar a `main` al final de tu archivo.

Pistas

- Recuerda que `input` devuelve un `str`, según docs.python.org/3/library/functions.html#input.
- Recuerda que un `str` viene con bastantes métodos, según docs.python.org/3/library/stdtypes.html#string-methods.
- Un emoji es en realidad sólo un carácter, así que puedes citarlo como cualquier `str`, como `"🙂"`. Y puedes copiar y pegar el emoji de esta página en tu propio código según sea necesario.

Demostración

```
$ python caras.py
Hola :)
Hola 🙂
$ python caras.py
Adios :(
Adios 🙁
$
```

Antes de Comenzar

Ejecuta:

mkdir caras

Para crear una carpeta llamada `caras` en tu espacio de código.

Después ejecuta:

cd caras

Para cambiar directorios a esa carpeta. Ahora deberías ver tu prompt de terminal como `caras/ $`.

Ahora puedes ejecutar:

code caras.py

Para crear un archivo llamado `caras.py` donde escribirás tu programa.

Cómo Probar

Aquí te mostramos cómo probar tu código manualmente:

Ejecuta tu programa con `python caras.py`. Escribe `Hola :)` y presiona Enter. Tu programa debería mostrar:

Hola 🙂

Ejecuta tu programa con `python caras.py`. Escribe `Adios :(` y presiona Enter. Tu programa debería mostrar:

Adios 🙁

Ejecuta tu programa con `python caras.py`. Escribe `Hola :) Adios :(` y presiona Enter. Tu programa debería mostrar:

Hola 🙂 Adios 🙁

Einstein

Incluso si no has estudiado física (recientemente o nunca!), podrías haber oído que **E = mc²**, donde **E** representa energía (medida en Joules), **m** representa masa (medida en kilogramos), y **c** representa la velocidad de la luz (medida aproximadamente como 300,000,000 metros por segundo), según Albert Einstein et al. Esencialmente, la fórmula significa que masa y energía son equivalentes.

En un archivo llamado `einstein.py`, implementa un programa en Python que solicite al usuario la masa como un entero (en kilogramos) y luego muestre el número equivalente de Joules como un entero. Asume que el usuario ingresará un entero.

Pistas

- Recuerda que `input` devuelve un `str`, según docs.python.org/3/library/functions.html#input.
- Recuerda que `int` puede convertir un `str` a un `int`, según docs.python.org/3/library/functions.html#int.
- Recuerda que Python viene con varias funciones integradas, según docs.python.org/3/library/functions.html.

Demostración

$ python einstein.py

m: 50

E: 4500000000000000000

$

Antes de Comenzar

Ejecuta:

mkdir einstein

para crear una carpeta llamada `einstein` en tu espacio de código.

Después ejecuta:

cd einstein

para cambiar directorios a esa carpeta. Ahora deberías ver tu prompt de terminal como `einstein/ $`.

Ahora puedes ejecutar:

code einstein.py

para crear un archivo llamado `einstein.py` donde escribirás tu programa.

Cómo Probar

Aquí te mostramos cómo probar tu código manualmente:

Ejecuta tu programa con `python einstein.py`. Escribe `1` y presiona Enter. Tu programa debería mostrar:

```
90000000000000000
```

Ejecuta tu programa con `python einstein.py`. Escribe `14` y presiona Enter. Tu programa debería mostrar:

```
1260000000000000000
```

Ejecuta tu programa con `python einstein.py`. Escribe `50` y presiona Enter. Tu programa debería mostrar:

```
4500000000000000000
```

Calculadora de propinas

> "Y ahora mi calculadora de propinas mágica." — Morty Seinfeld

En los Estados Unidos, es costumbre dejar una propina para tu mesero después de cenar en un restaurante, típicamente una cantidad igual al 15% o más del costo de tu comida. ¡No te preocupes, hemos escrito una calculadora de propinas para ti, abajo!

```
def main():
    dolares = dolar_a_float(input("Cuánto costó la comida? "))
    porcentaje = porcentaje_a_float(input("Qué porcentaje te gustaría dejar de propina? "))
    propina = dolares * porcentaje
    print(f"Dejar ${propina:.2f}")

def dolar_a_float(d):
    # TODO

def porcentaje_a_float(p):
    # TODO

main()
```

Bueno, hemos escrito la mayor parte de una calculadora de propinas para ti. Desafortunadamente, no tuvimos tiempo de implementar dos funciones:

- `dolar_a_float`, que debería aceptar un `str` como entrada (formateado como `$##.##`, donde cada `#` es un dígito decimal), eliminar el `$` inicial, y devolver la cantidad como un `float`. Por ejemplo, dado `$50.00` como entrada, debería devolver `50.0`.

- `porcentaje_a_float`, que debería aceptar un `str` como entrada (formateado como `##%`, donde cada `#` es un dígito decimal), eliminar el `%` final, y devolver el porcentaje como un `float`. Por ejemplo, dado `15%` como entrada, debería devolver `0.15`.

Asume que el usuario ingresará valores en los formatos esperados.

Pistas

- Recuerda que `input` devuelve un `str`, según docs.python.org/3/library/functions.html#input.
- Recuerda que `float` puede convertir un `str` a un `float`, según docs.python.org/3/library/functions.html#float.
- Recuerda que un `str` viene con bastantes métodos, según docs.python.org/3/library/stdtypes.html#string-methods.

Demostración

$ python propina.py
Cuánto costó la comida? $50.00
Qué porcentaje te gustaría dejar de propina? 15%
Dejar $7.50

Antes de Comenzar

Luego ejecuta:

mkdir propina

para crear una carpeta llamada `propina` en tu espacio de código.

Después ejecuta:

cd propina

para cambiar directorios a esa carpeta. Ahora deberías ver tu prompt de terminal como `propina/ $`.

Ahora puedes ejecutar:

code propina.py

para crear un archivo llamado `propina.py`. Copia y pega el código de arriba en un archivo, y completa las implementaciones de `dolar_a_float` y `porcentaje_a_float`, reemplazando cada `TODO` con una o más líneas de tu propio código.

Cómo Probar

Aquí te mostramos cómo probar tu código manualmente:

Ejecuta tu programa con `python propina.py`. Escribe `$50.00` y presiona Enter. Luego, escribe `15%` y presiona Enter. Tu programa debería mostrar:

```
Dejar $7.50
```

Ejecuta tu programa con `python propina.py`. Escribe `$100.00` y presiona Enter. Luego, escribe `18%` y presiona Enter. Tu programa debería mostrar:

```
Dejar $18.00
```

Ejecuta tu programa con `python propina.py`. Escribe `$15.00` y presiona Enter. Luego, escribe `25%` y presiona Enter. Tu programa debería mostrar:

```
Dejar $3.75
```