Formulen un programa que:  
**i. A una variable se le asigne un mensaje motivador que incluya los nombres de todos los integrantes. ¿Qué tipo de dato puede ser?**

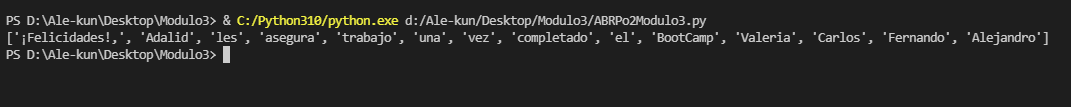
**ii. Se asegure que todos su caracteres estén en mayúscula.**

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamenteTexto

Descripción generada automáticamenteEl dato es de tipo lista (en donde está cada uno de los integrantes), que se vuelve string gracias al método .join() y puede ser devuelta como texto.

Para dejar todos los caracteres en mayúscula utilizamos el método .upper() que devuelve una cadena donde todos los caracteres están en mayúscula, e ignora lo símbolos y números.

Texto

Descripción generada automáticamente  
**iii. Elabore una lista con cada palabra del string.**

Texto

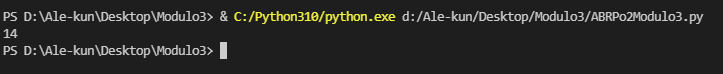
Descripción generada automáticamente

**iv. Cada integrante del grupo debe reconocer en qué índice está su nombre.**

Texto

Descripción generada automáticamente

**v. Indique cuántas palabras tenía el string.**

Para indicar la cantidad de palabras primero desarmamos la lista en strings y la almacenamos en la variable “x”, le sumamos el mensaje de la variable “y”, además de volver todo en mayúscula, todo fue guardado en la variable “w”.

Finalmente utilizamos el método .split() para separar cada una de las palabras y las contamos con len. Lo que nos entregó un total de 14 palabras.

**vi. Imprima una tupla con todas las palabras del string.**

Texto

Descripción generada automáticamenteComo se lee en los elementos comentados, la variable “q” que se mostró en la terminal, es la suma de las dos listas antes desarrolladas (una original -integrantes, y la otra a partir del ejercicio hecho en la variable “z”) mediante la función “tuple()” , la convertimos en una única tupla.

**vii. ¿Con qué función podrían obtener el mensaje por teclado al ejecutar el programa? Implementarlo!.**

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza mediaTexto

Descripción generada automáticamente

- Discutan ¿Qué es un dato booleano? ¿Qué utilidad puede tener para el desarrollo de un programa?  
- Investigar qué significa que python sea un lenguaje de tipado dinámico.  
- Investigar y documentar sobre la creación de Módulos en Python.  
- Investigar y documentar sobre la creación de Paquetes en Python.  
- Investigar e implementar el uso del archivo \_\_init\_\_.py

Nombre del proyecto:  
“Te lo Vendo” – Iteración 2

Discutan ¿Qué es un dato booleano? ¿Qué utilidad puede tener para el desarrollo de un programa?

Un dato booleano es un dato lógico, que puede tener como valor *verdadero* o *falso*¸ True o False respectivamente. Su valor por defecto es False. Todos los demás valores, incluido cualquier objeto, un arreglo vacío ([]) o la cadena "false", crean un objeto con un valor inicial de true.

Su uso suele ser mediante operadores lógicos (AND, OR, NOT), que, al relacionarse conforman lo que se conoce como *Truth Table*, o tabla de verdad:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | A AND B | A OR B | NOT A |
| False | False | False | False | True |
| False | True | False | True | True |
| True | False | False | True | False |
| True | True | True | True | False |

#### Investigar y documentar sobre la creación de Módulos en Python.

#### Investigar qué significa que python sea un lenguaje de tipado dinámico. un lenguaje de programación tiene un sistema de tipo dinamico cuando el tipo de dato es una variable que puede cambiar durante la ejecución. De esta forma, una variable puede comenzar teniendo un tipo de dato y cambia en cualquier momento a otro tipo de dato. En el siguiente ejemplo, una variable puede ser inicialmente del tipo numero entero y cambiar a string 1. a = 5 2. print(a) 3. a = "Hola mundo" 4. print(a) Aquí la variable a es creada con el valor 5, que es un número entero (int). Luego en la tercera línea se asigna el nuevo valor "Hola mundo", por lo cual el tipo de dato cambia a una cadena (str).

#### **Los módulos en Python**

#### Cuando creas programas en Python, los archivos generados suelen tener la extensión particular de Python, ya sabes, .py. Los módulos en Python se crean de la misma manera. Es decir, son archivos de código con extensión .py, y ubicados en un directorio donde Python sea capaz de encontrarlos (es decir, el directorio de trabajo actual o listados en sys.path).

#### Los módulos contienen normalmente sentencias que están relacionadas entre sí. Como he mencionado anteriormente, podemos utilizar módulos en cualquier momento. El uso de un módulo consiste en utilizar el código (es decir, variables, funciones) almacenado en dicho módulo. El proceso de añadir y utilizar dicho código se llama importación.

#### Creación de módulos

#### Crear un módulo en Python es muy simple. Digamos que queremos crear un módulo que imprima el nombre de alguien. Escribe el siguiente código utilizando tu editor favorito, y guardalo como myname.py. Esta será el nombre de tu módulo, excluyendo la parte .py, el cual se asignará a la variable global \_\_name\_\_.

#### def print\_name(n):     print 'Hola', n

#### Importando módulos

#### Si tienes un archivo de Python y quieres añadir el código del módulo que hemos creado anteriormente, lo importaremos utilizando la palabra clave import, tal que así:

#### import myname name.print\_name('Jorge')

#### El código de Python para un módulo llamado aname normalmente reside en un archivo llamado aname.py. Aquí hay un ejemplo de un módulo simple, support.py

#### def print\_func (par):

#### print "Hola:", par

#### return

#### Localizando módulos

#### Cuando usted importa un módulo, el interprete Python busca por el módulo en la secuencia siguiente:

#### El directorio actual.

#### Si el módulo no es encontrado, Python entonces busca en cada directorio en la variable de entorno [PYTHONPATH](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion8/modulos.html#python-variable-entorno-path) del sistema operativo.

#### Si todas las anteriores fallan, Python busca la ruta predeterminada. En UNIX, la ruta predeterminada normalmente esta /usr/local/lib/python/.

#### El ruta de búsqueda de módulo es almacenado en el módulo de system sys como la variable sys.path. La variable sys.path contiene el directorio actual, PYTHONPATH, y las predeterminadas dependencia de instalación.

#### PYTHONPATH[**¶**](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion8/modulos.html#pythonpath)

#### Es una variable de entorno del sistema operativo, consistiendo de una lista de directorios. La sintaxis de PYTHONPATH es la misma como la del Shell de la variable PATH.

#### Así es una típica definición de PYTHONPATH desde un sistema Windows, ejecutando:

#### set PYTHONPATH = C:\python20\lib;

#### Así es una típica definición de PYTHONPATH desde un sistema UNIX, ejecutando:

#### set PYTHONPATH = /usr/local/lib/python

#### material anexo en el siguiente link :

#### <https://www.youtube.com/watch?v=t93x-vnFvP4>

#### Investigar y documentar sobre la creación de paquetes en Python

#### Además de la versión de Python que se esté utilizando, es necesario tener la posibilidad de instalar paquetes no estándar de Python con versiones específicas: para instalar paquetes para un proyecto en particular de Python, se debe utilizar la herramienta PIP (Python Package Index) que es el sistema de gestión de paquetes de Python. Para la instalación se debe tener activado el ambiente virtual.

#### Entre los paquetes de Python más conocidos se tienen a:

#### Urllib3

#### Six

#### Botocore

#### Pip

#### Requests

#### Certifi

#### Idna

#### PyYAML

#### Pyasn1

#### Docutils

#### Chardet

#### RSA

#### Beautifulsoup4

#### Estos paquetes facilitan el desarrollo de un proyecto Python. Por ejemplo, Requests facilita el manejo de mensajes del protocolo HTTP y Beautifulsoup4 que facilita la extracción de datos de archivos HTMl y XML.

#### Investigar e implementar el uso del archivo \_\_init\_\_.py

#### ¿Para qué sirve \_\_init\_\_.py?

#### Cuando recién empezaba a crear sitios web con Django y Python, se me hacia raro encontrar archivos de tipo init.py dentro de las carpetas. Si tu tienes dudas sobre este archivo, en este tutorial te explicare para que sirve.

#### ¿Para qué se usa el archivo init.py?

#### Bueno, este archivo sirve para que Python identifique a la carpeta que lo contiene como un directorio de paquetes de Python. De tal manera que podamos importar los módulos que contiene esa carpeta.

#### Por ejemplo tenemos el directorio operaciones y dentro tenemos dos archivos suma.py y resta.py. Si agregamos el archivo init.py dentro de la carpeta operaciones. Entonces luego podremos hacer lo siguiente:

#### from operaciones import suma, resta

#### o

#### import operaciones.suma

#### No necesitas agregar nada dentro del archivo init.py, pero si deseas puedes inicializar partes de los paquetes que contiene el directorio.

#### Si remueves el archivo init.py, no podrás importar los paquetes que contiene el directorio, como vimos en el ejemplo de más arriba. Y te saldrá un mensaje de error parecido a esto:

#### ImportError: No module named operaciones

#### Este archivo es muy importante a la hora de trabajar en Python, ya que nos permitirá organizar mejor nuestro código. Así que ya sabes para que funciona este archivo.

#### Nota de referencia :

#### https://pythondiario.com/2013/06/initpy-en-python-que-es-y-como-funciona.html

#### Los módulos en Python

#### Cuando creas programas en Python, los archivos generados suelen tener la extensión particular de Python, ya sabes, .py. Los módulos en Python se crean de la misma manera. Es decir, son archivos de código con extensión .py, y ubicados en un directorio donde Python sea capaz de encontrarlos (es decir, el directorio de trabajo actual o listados en sys.path).

#### Los módulos contienen normalmente sentencias que están relacionadas entre sí. Como he mencionado anteriormente, podemos utilizar módulos en cualquier momento. El uso de un módulo consiste en utilizar el código (es decir, variables, funciones) almacenado en dicho módulo. El proceso de añadir y utilizar dicho código se llama importación.

#### Creación de módulos

#### Crear un módulo en Python es muy simple. Digamos que queremos crear un módulo que imprima el nombre de alguien. Escribe el siguiente código utilizando tu editor favorito, y guardalo como myname.py. Esta será el nombre de tu módulo, excluyendo la parte .py, el cual se asignará a la variable global \_\_name\_\_.

#### def print\_name(n):     print 'Hola', n

#### Importando módulos

#### Si tienes un archivo de Python y quieres añadir el código del módulo que hemos creado anteriormente, lo importaremos utilizando la palabra clave import, tal que así:

#### import myname name.print\_name('Jorge')

#### El código de Python para un módulo llamado aname normalmente reside en un archivo llamado aname.py. Aquí hay un ejemplo de un módulo simple, support.py

#### def print\_func (par):

#### print "Hola:", par

#### return

#### Localizando módulos

#### Cuando usted importa un módulo, el interprete Python busca por el módulo en la secuencia siguiente:

#### El directorio actual.

#### Si el módulo no es encontrado, Python entonces busca en cada directorio en la variable de entorno [PYTHONPATH](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion8/modulos.html#python-variable-entorno-path) del sistema operativo.

#### Si todas las anteriores fallan, Python busca la ruta predeterminada. En UNIX, la ruta predeterminada normalmente esta /usr/local/lib/python/.

#### El ruta de búsqueda de módulo es almacenado en el módulo de system sys como la variable sys.path. La variable sys.path contiene el directorio actual, PYTHONPATH, y las predeterminadas dependencia de instalación.

#### PYTHONPATH[**¶**](https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion8/modulos.html#pythonpath)

#### Es una variable de entorno del sistema operativo, consistiendo de una lista de directorios. La sintaxis de PYTHONPATH es la misma como la del Shell de la variable PATH.

#### Así es una típica definición de PYTHONPATH desde un sistema Windows, ejecutando:

#### set PYTHONPATH = C:\python20\lib;

#### Así es una típica definición de PYTHONPATH desde un sistema UNIX, ejecutando:

#### set PYTHONPATH = /usr/local/lib/python

#### material anexo en el siguiente link :

#### <https://www.youtube.com/watch?v=t93x-vnFvP4>

#### Investigar y documentar sobre la creación de paquetes en Python

#### Además de la versión de Python que se esté utilizando, es necesario tener la posibilidad de instalar paquetes no estándar de Python con versiones específicas: para instalar paquetes para un proyecto en particular de Python, se debe utilizar la herramienta PIP (Python Package Index) que es el sistema de gestión de paquetes de Python. Para la instalación se debe tener activado el ambiente virtual.

#### Entre los paquetes de Python más conocidos se tienen a:

#### Urllib3

#### Six

#### Botocore

#### Pip

#### Requests

#### Certifi

#### Idna

#### PyYAML

#### Pyasn1

#### Docutils

#### Chardet

#### RSA

#### Beautifulsoup4

#### Estos paquetes facilitan el desarrollo de un proyecto Python. Por ejemplo, Requests facilita el manejo de mensajes del protocolo HTTP y Beautifulsoup4 que facilita la extracción de datos de archivos HTMl y XML.

#### Investigar e implementar el uso del archivo \_\_init\_\_.py

#### ¿Para qué sirve \_\_init\_\_.py?

#### Cuando recién empezaba a crear sitios web con Django y Python, se me hacia raro encontrar archivos de tipo init.py dentro de las carpetas. Si tu tienes dudas sobre este archivo, en este tutorial te explicare para que sirve.

#### ¿Para qué se usa el archivo init.py?

#### Bueno, este archivo sirve para que Python identifique a la carpeta que lo contiene como un directorio de paquetes de Python. De tal manera que podamos importar los módulos que contiene esa carpeta.

#### Por ejemplo tenemos el directorio operaciones y dentro tenemos dos archivos suma.py y resta.py. Si agregamos el archivo init.py dentro de la carpeta operaciones. Entonces luego podremos hacer lo siguiente:

#### from operaciones import suma, resta

#### o

#### import operaciones.suma

#### No necesitas agregar nada dentro del archivo init.py, pero si deseas puedes inicializar partes de los paquetes que contiene el directorio.

#### 