UNIVERSIDAD DIEGO PORTALES

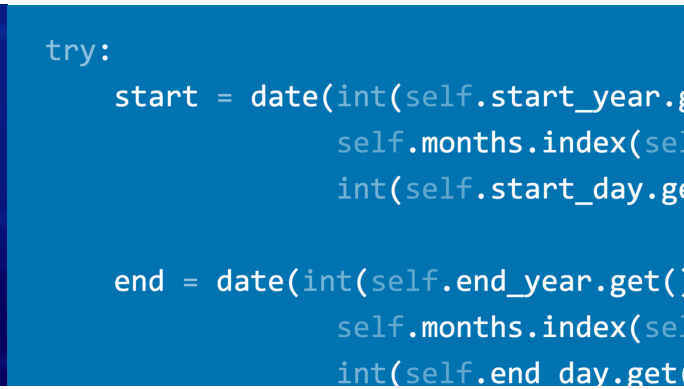
Introducción a Python

Introducción

2026

Temario

- Estructuras de control
- Listas
- Funciones integradas
  - Matemáticas



Hasta el momento hemos utilizado funciones básicas en nuestros programas:

```
print("texto ",var)
```

función que recibe múltiples parámetros y lo despliega por pantalla, no retorna ningún valor.

```
int(var)
```

función que recibe un valor, y retorna el mismo valor transformado en un entero

```
input("texto")
```

función que recibe un parámetro, lo despliega por pantalla, recibe un valor desde el teclado y retorna un texto.

Sin embargo, existen miles de módulos en Python disponibles para ser usados por programadores en sus programas.

Para hacer uso de una función de la biblioteca, se llama a la biblioteca y la función correspondiente

Importando módulos

```
import <nombreMódulo>
nombreMódulo.nombreFunción(<parametros>)
```

https://i.makeagif.com/media/3-19-2019/wDW4V-.mp4

Ayuda en Python

Para obtener información de una biblioteca en particular, por ejemplo, las funciones existentes, vaya al terminal, importe el módulo y llame a la función: **help(<nombre módulo>)**

Ejemplo:
import math
help(math)

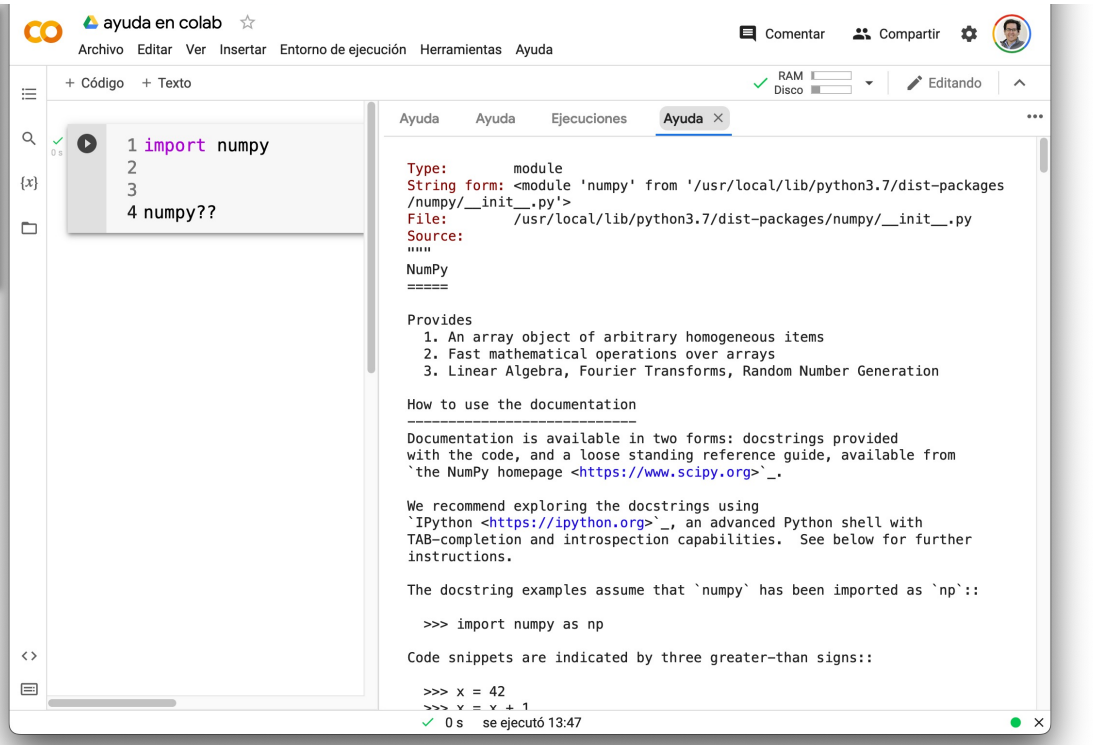
```
>>> help(math)
Help on module math:

NAME
    math

MODULE REFERENCE
    https://docs.python.org/3.6/library/math

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(...)
        acos(x)

        Return the arc cosine (measured in radians) of x.
```

Para obtener información de una biblioteca en particular, por ejemplo, las funciones existentes, vaya al terminal, importe el módulo y llame a la función: **<nombre módulo>??**

Google colab

Ejemplo:
import numpy

numpy??



Dentro de los módulos más útiles en Python podemos encontrar:

- math para operaciones matemáticas
- random para generar números aleatorios
- matplotlib para graficar
- numpy y pandas para análisis de datos

- Y muchos más 😎

Módulo math

Veamos algunas funciones importantes del módulo **math**

- **math.ceil(x)** : retorna el entero más pequeno que sea igual o mayor que *x*
- **math.exp(x)** : retorna el valor exponencial de x
- **math.fabs(x)** : retorna el valor absoluto de x
- **math.factorial(x)** : retorna el factorial de x
- **math.floor(x)** : retorna el entero más grande que sea igual o menor que *x*
- **math.log**(x,baseLog) : retorna el logaritmo de x en base baseLog
- **math.pow(x,y)** : retorna x elevado y
- **math.sqrt(x)** : retorna la raiz cuadrada de x

Tiempo : 10 minutos

Cree un programa que le solicite al usuario un número z y un número de iteraciones numIter por pantalla y calcule la sumatoria:

$$\sum_{k=0}^{numIter} \frac{z^k}{k!}$$

math.factorial(x) : retorna el factorial de x:

math.pow(x,y) : retorna x elevado y

importamos el módulo

```
import math
z = int(input("ingrese numero: "))
numIter = int(input("ingrese numero iteraciones: "))
suma = 0
for k in range(0,numIter+1):
        suma = suma + math.pow(z,k) / math.factorial(k)
print("el valor es ",suma)
```

Veamos algunas funciones importantes del módulo **time**

time.sleep(x) : el programa se pausa por x segundos

time.time() : devuelve el número de segundos desde el 1 de Enero de 1970 (año que se utiliza como valor inicial en Unix)

time.ctime(x) : convierte el tiempo expresado en segundos en el formato de la fecha actual: Día de la semana, Mes, día, hora, año.

Veamos algunas funciones importantes de la biblioteca **random**

random.randint (x,y) : retorna un número aleatorio entero entre x e y.

random.random () : retorna un número aleatorio entre 0 y 1.

random.uniform (x,y) : retorna un número aleatorio uniforme entre x e y.

Tiempo : 10 minutos

Cree un programa que genere una multiplicación entre dos números en forma aleatoria y le pida al usuario resolverla. Si el usuario se demoró más de 10 segundos en resolverla, despliegue el mensaje "eres muy lento, el resultado era X". Caso contrario, verifique si el usuario pudo resolver la multiplicación en forma correcta, felicitándolo o mostrando la solución de la multiplicación.

time.time( ): devuelve el número de segundos desde el 1 de Enero de 1970 (año que se utiliza como valor inicial en Unix)

random.randint (x,y): retorna un número aleatorio entero entre x e y.

```
import time
import random

time1 = time.time()
num1  = random.randint(1,10)
num2  = random.randint(1,10)
print("¿Cuánto es ",num1," por ",num2,"? ",end="")
res = int(input())
time2 = time.time()

if time2-time1>10:
    print("Eres muy lento, el resultado era ",num1*num2)
else:
    if num1*num2==res:
        print("FELICITACIONES")
    else:
        print("Te equivocaste, el resultado era ",num1*num2)
```

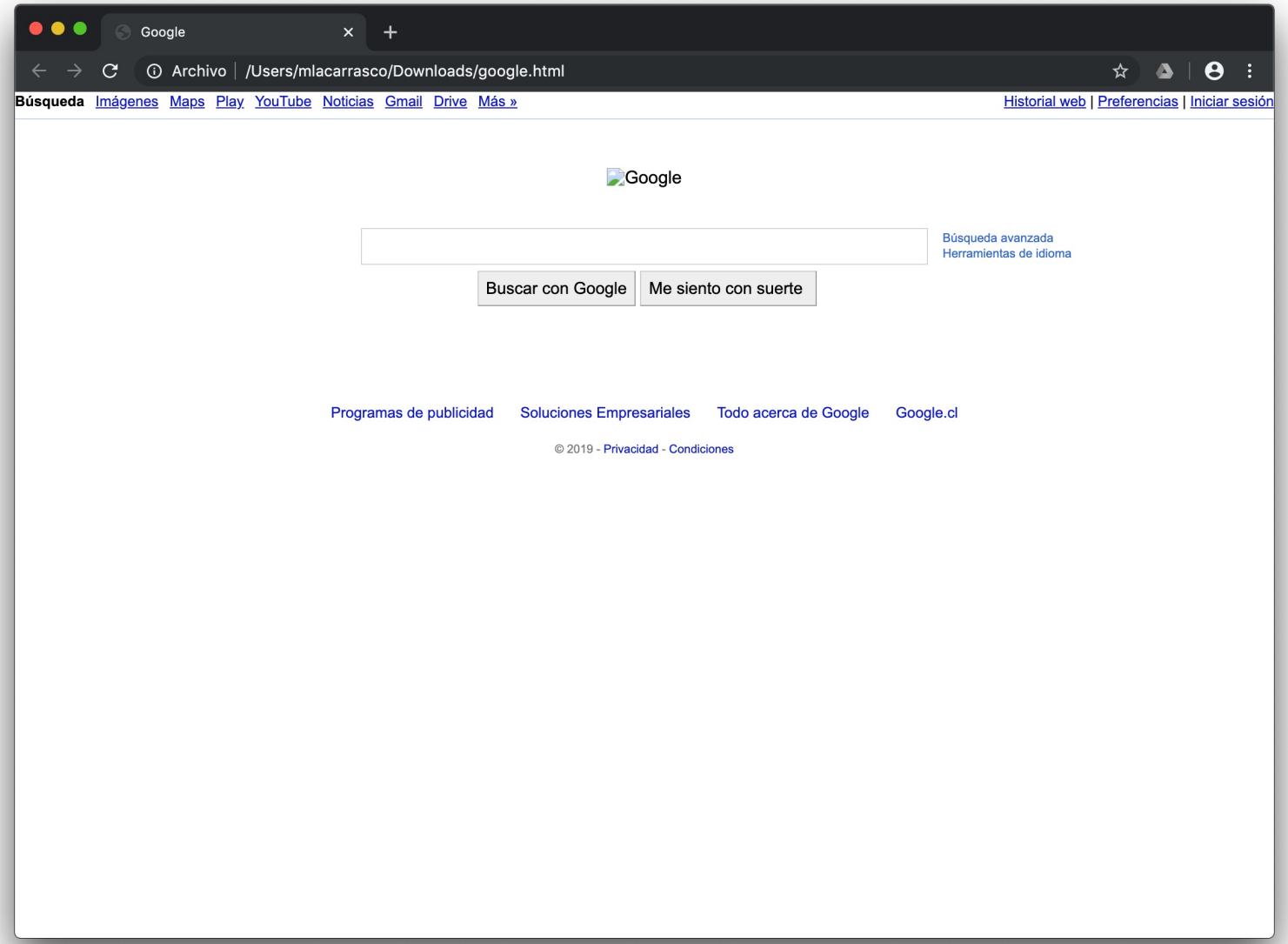
Módulo request

Un módulo que se usa bastante es **request.**
Lo pueden ver como un navegador web simple

```
import requests

x = requests.get('http://www.google.cl')
print(x.text)
```

Copia el resultado y guárdalo en un archivo llamado google.html
Ahora abre el archivo con Chrome u otro navegador…¿qué ven?

"Google"


Google
Archivo | /Users/mlacarrasco/Downloads/google.html
Búsqueda Imágenes Maps Play YouTube Noticias Gmail Drive Más »
Historial web | Preferencias | Iniciar sesión
Google
Búsqueda avanzada
Herramientas de idioma
Buscar con Google
Me siento con suerte
Programas de publicidad
Soluciones Empresariales
Todo acerca de Google
Google.cl
© 2019 - Privacidad - Condiciones

Módulo request

Para automatizar el proceso de guardar a un archivo, podemos crear uno automáticamente y guardar esta información en él
Lo pueden ver como un navegador web simple

```
import requests
file = open("emol.html","w")

x = requests.get('http://www.emol.com')

print(x.text)
file.write(x.text)
file.close()
```

Creamos un archivo en modo escritura que llamamos emol.html
"w": modo escritura

otros modos

"a": append, agrega al final del archivo

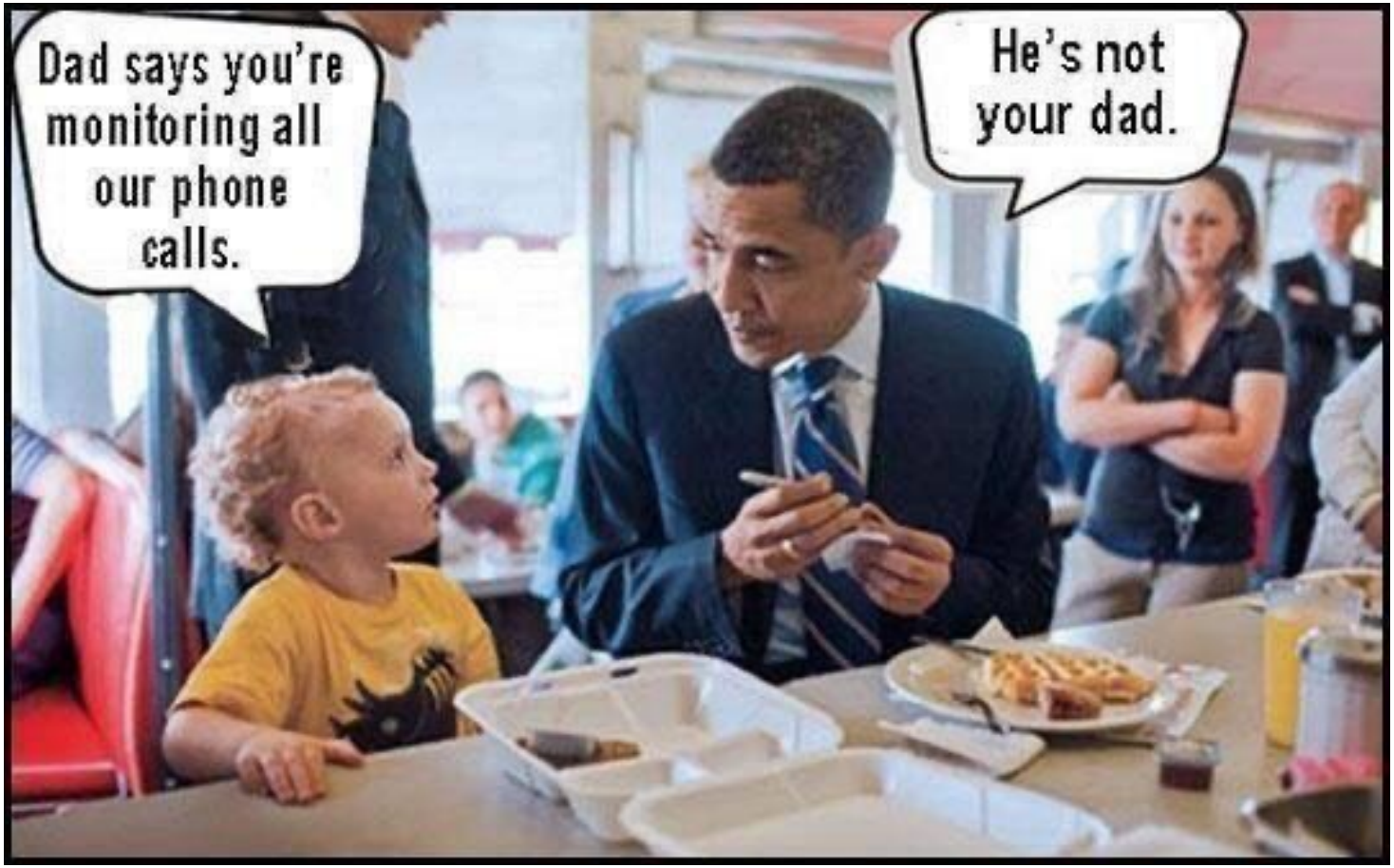
Luego de imprimir en el arcchivo, escribimos en él, y luego lo cerramos

Ejercicio

Tiempo : 5 minutos

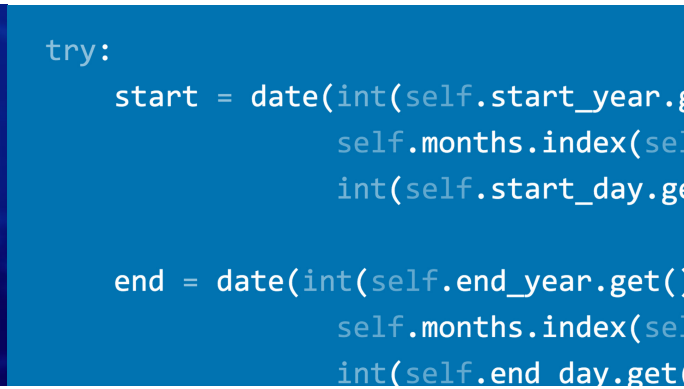
Usa el módulo requests para obtener el contenido de una página que te indique el usuario…será el inicio de tu navegador ☺, y guadala en un archivo

```
import requests

namefile = input("Ingresa el nombre del archivo: ")
file = open(namefile,"w")

url = input("Ingresa el nombre de la URL: ")
x = requests.get(url)

print(x.text)
file.write(x.text)
file.close()
```


Dad says you're monitoring all our phone calls.
He's not your dad.

Temario

- Estructuras de control
- Listas
- Funciones integradas
  - Matemáticas
  - Numpy



UNIVERSIDAD DIEGO PORTALES

Recordemos Ya hemos visto en Python el concepto de listas

- Una gran “cajonera” donde podemos almacenar distintos valores
- Podemos acceder a los valores usando su posición en la lista
- Una lista puede contener distintos tipos de datos: texto, números, booleanos, entre otros

Python

```
xs = [3, 1, 2]                  # crea una lista
print(xs, xs[2])                # muestra en pantalla [3, 1, 2] 2
xs[2] = 'hola'                  # listas pueden tener distintos tipos de datos
print(xs)                       # muestra en pantalla [3, 1, ‘hola']
xs.append('chao')                       # agrega un elemento al final de la
lista
print(xs)                       # muestra en pantalla [3, 1, ‘hola', ‘chao']
```

Realice un programa que cree una lista con 10 elementos generados al azar y muestre en pantalla el mayor de ellos

```
import random

lista = []
for i in range(10):
    lista.append(random.random())

print(lista)

mayor = lista[0]

for ele in lista:
    if ele>=mayor:
        mayor = ele

print("El mayor es:", mayor)
```

Problema

Al ser una lista algo tan flexible, se generaliza su uso.

- No puedo realizar las mismas acciones sobre todos los elementos pues algunos pueden ser de distinto tipo
- No puedo ir a una ubicación específica en dos dimensiones en forma inmediata, pues no existe el concepto de tamaño fijo de una lista.

Por ello presentamos ….

It's so easy!

Numpy

Es un módulo en Python que agrega soporte para arreglos en una o múltiples dimensiones (como matrices), y una gran colección de operaciones matemáticas para operar sobre estos arreglos

Importamos la biblioteca numpy

```
import numpy

xs = numpy.array([3, 1, 2])
print(xs)
```

Creamos un arreglo con números

¿Notaron que

… el arreglo en NumPy no separa los valores con una coma al imprimirlo en pantalla?

`array()`

Los arreglos son el equivalente a una lista en Python, en el sentido que almacena un conjunto de valores. Se crean usando la función array de numpy.

```
import numpy

arreglo_primero = numpy.array([3, 1, 2])
arreglo_segundo = numpy.array([3, 'hola', 2])
arreglo_tercero = numpy.array([3.21, 1, 2])

print(arreglo_primero)
print(arreglo_segundo)
print(arreglo_tercero)
```

El primer arreglo contiene solamente números, sin embargo el segundo, al tener un texto, convierte todos los elementos a texto, y el tercero a decimales

Importante

Los arreglos NO permite mezclar distintos tipos de datos, y solo acepta valores de un mismo tipo convirtiendo a un tipo común.

¿Para qué sirve?

Realice un programa que cree una lista con 10 elementos generados al azar y muestre en pantalla el mayor de ellos

```
import random

lista = []
for i in range(10):
    lista.append(random.random())

print(lista)

mayor = lista[0]

for ele in lista:
    if ele>=mayor:
        mayor = ele

print("El mayor es:", mayor)
```

¿Para qué sirve?

Realice un programa que cree un arreglo con 10 elementos generados al azar y muestre en pantalla el mayor de ellos

```
import random
import numpy

arreglo = numpy.random.random(10)
print("el maximo es: ", arreglo.max())
```

Numpy provee muchas alternativas para crear arreglos:

numpy.zeros(dimensión): Crea un arreglo de tamaño **dimensión** y le asigna a cada elemento valor **0**

```
a = numpy.zeros(5,)
print(a) # imprime [0. 0. 0. 0. 0.]
```

numpy.ones(dimensión): Crea un arreglo de tamaño **dimensión** y le asigna a cada elemento valor **1**

```
b = numpy.ones((2,))
print(b)
```

numpy.full(dimensión, valor): Crea un arreglo de tamaño **dimensión** y le asigna a cada elemento valor **valor**

```
c = numpy.full((2,), 7)
print(c)          # muestra [7 7]
```

numpy.random.random(dimensión): Crea un arreglo de tamaño **dimensión** y le asigna a cada elemento valor **random**

```
d = numpy.random.random(5,)
print(d)
```

Accediendo a elementos

Al igual que con listas, se puede recuperar el elemento en una posición usando la notación

```
arreglo[posición]
```

Ejemplo:

```
import numpy

a = numpy.random.random(5,)

print(a[3])
```

Agregando elementos

Para agregar elementos a un arreglo tenemos la función append() de la biblioteca Numpy

```
import numpy

a = numpy.array([1,2,3,4,5,6])

print(a)

b = numpy.append(a,[7])
print(a)
print(b)
```

Nota que él o los elementos a agregar deben ser listas o arreglos

Importante

Los arreglos NO se actualizan cuando usamos append, sino que se crea una copia del arreglo con él o los elementos agregados.

Tamaño de un arreglo

Para obtener el tamaño de un arreglo podemos usar len(arreglo), tal como con listas, o el atributo llamado size

```
import numpy

a = numpy.random.random(5,)

print(a.size, "vs", len(a))
```

Recorrer un arreglo

Para recorrer un arreglo, podemos utilizar los mismos métodos que con listas:

```
import numpy

a= numpy.array([1,2,4,8,16,32])

for i in a:
   print(i)

for i in range(a.size):
   print(a[i])
```

Ordenar un arreglo

Para ordenar un arreglo, a través de los índices del arreglo empleamos el método argsort(). De esta forma siempre podemos volver a los datos originales sin orden.

```
import numpy as np

data = np.random.randint(0,10,[10,])
print(data)

index_sort = np.argsort(data)

print(data[index_sort])
```

Ordenar un arreglo

Para ordenar un arreglo, a través de los índices del arreglo empleamos el método argsort(). De esta forma siempre podemos volver a los datos originales sin orden.

```
import numpy as np

data = np.random.random_integers(0,20,(10,2))

index_sort = np.argsort(data, axis=0)

print(data[index_sort[:,1])
```

Indicamos las filas y columnas de nuestra matriz

Indicamos si ordenamos por columnas (0), o por filas (1)

Indicamos la columna que será ordenada

Sin embargo, lo realmente entretenido de numpy son los cálculos sobre los arreglos. Por ejemplo:

`arreglo*num` : retorna un arreglo donde todos los elementos son multiplicados por num (pueden usar /, +, -)

`arreglo.min()` : retorna el valor mínimo de un arreglo

`arreglo.round(decimales)` : retorna un arreglo con todos los elementos redondeados a la cantidad de decimales pasados

`arreglo.sum()` : retorna la suma de los elementos de un arreglo

`arreglo.mean()` : retorna el promedio de los elementos de un arreglo

`arreglo.prod()` : retorna la multiplicación de los elementos de un arreglo

Escriba un programa en Python que, para un **arreglo** con las notas de un curso, calcule el promedio de notas del curso, extraiga la menor y la mayor nota, para luego mostrar estas tres cosas por pantalla.

arreglo de notas

```
[3.4, 4.3, 4.7, 5.1, 5.3, 5.8, 6.1, 6.7, 7.0]
```

Escriba un programa en Python que, para un **arreglo** con las notas de un curso, calcule el promedio de notas del curso, extraiga la menor y la mayor nota, para luego mostrar estas tres cosas por pantalla.

```
import numpy as np

notas = np.array([3.4, 4.3, 4.7, 5.1, 5.3, 5.8, 6.1, 6.7, 7.0])

promedio = notas.mean()
menor = notas.min()
mayor = notas.max()

print('El promedio es', promedio)
print('La nota más baja es', menor)
print('La nota más alta es', mayor)
```

Ejemplo

```
import numpy as np

a = np.array([1,2,4,8,16,32])

print (a*4)

print('La suma de los elementos de a es ', a.sum())
print('y su multiplcación es', a.prod())
print('El promedio de los elementos es', a.mean())
```

Si los arreglos tienen igual tamaño entonces puedo realizar operaciones de suma, multiplicación, resta, división

```
import numpy as np

a = np.array([1,2,4,8,16,32])
b = np.array([0,1,1,2,2,3])


print (a+b)
print (a*b)
```

Un científico mexicano estaba haciendo estudios de la altura de las personas en nuestro país, y guardó sus datos en un arreglo numpy llamado alturas. Sin embargo, utilizó pulgadas como medida de altura

Escriba el código para transformar esas mediciones en centímetros

Un científico mexicano estaba haciendo estudios de la altura de las personas en nuestro país, y guardó sus datos en un arreglo numpy llamado alturas. Sin embargo, utilizó pulgadas como medida de altura

Escriba el código para transformar esas mediciones en centímetros

```
import numpy as np

alturas = np.array([59.2, 73., 45.9, 65.7, 80.1])
en_centimetros = alturas * 2.54

print(en_centimetros)
```

Ahora tú

Suponga que usted tiene las dos evaluaciones del curso de Python para análisis de datos. Usted tiene la tarea de encontrar el promedio de cada estudiante, y buscar aquel estudiante que tenga la mejor calificación promedio.

Debido a que los arreglos de numpy son indexados desde la posición inicial 0, estos pueden ser manipulados por sus índices.

```
import numpy as np

a = np.array([1,2,4,8,16,32])

b = a[0:3]
c = a[:3]
d = a[3:]
e = a[:-1]
f = a[::-1]
```

- `b = a[0:3]` → Desde el índice 0 al índice 2 (0,1,2)
- `c = a[:3]` → Desde el inicio al índice 2 (0,1,2)
- `d = a[3:]` → Desde el índice 3 al final
- `e = a[:-1]` → Desde el inicio hasta uno menos el final
- `f = a[::-1]` → De inicio al final, pero con índices en decremento en -1

Incluso podemos usar los arreglos como valores de verdad usando un par de funciones:

`arreglo.all()` : retorna verdadero si todos los elementos de un arreglo se evaluan como verdadero

`arreglo.any()` : retorna verdadero si alguno de los elementos de un arreglo se evaluan como verdadero

```
import numpy

a= numpy.array([False, False, False, True, False])
b= numpy.array([1, 1, 1, 1, 1, 1, 1])

print(a.any())
print(a.all())

print(b.any())
print(b.all())
```

Escriba un programa en Python que, para un **arreglo** con las notas de un curso, calcule el promedio de las notas mayores a 5.5.

arreglo de notas

```
[3.4, 4.3, 4.7, 5.1, 5.3, 5.8, 6.1, 6.7, 7.0]
```

When you replace a for loop with a vectorized numpy function and see the speed improvement