



UAI
UNIVERSIDAD ADOLFO IBÁÑEZ

PROGRAMACIÓN ^{tics100}

FACULTAD DE INGENIERÍA Y CIENCIAS.
UNIVERSIDAD ADOLFO IBÁÑEZ

Numpy

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')              # Add a new element to the end of the list
print(xs)                     # muestra en pantalla [3, 1, 'hola', 'chao']
```

Problema

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

Por ejemplo:

- 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.

Entre otras cosas

¡NumPy al rescate!

¿Qué es?

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.

Python

```
import numpy                # importamos la biblioteca NumPy  
  
xs = numpy.array([3, 1, 2]) # creamos un arreglo de números  
print(xs)                  # muestra en pantalla [3 1 2]
```



▼ ¿Notaron que ...

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

Arreglos

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.

▼ repl.it

main.py

```
1 import numpy
2
3 primer_arreglo = numpy.array([3, 2, 1])
4 segundo_arreglo = numpy.array([3, 'hola', 2])
5 tercer_arreglo = numpy.array([3.2, 2, 1])
6
7 print(primer_arreglo)
8 print(segundo_arreglo)
9 print(tercer_arreglo)
10
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
[3 2 1]
['3' 'hola' '2']
[3.2 2.  1.]
>
```

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

¿Para qué sirve?



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

saved share run

main.py

```
1 import random
2
3 lista = []
4 for i in range(10):
5     lista.append(random.random())
6
7 mayor = 0
8 for i in lista:
9     if i > mayor:
10         mayor = i
11
12 print('El mayor elemento es', mayor)
```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
✦
El mayor elemento es 0.919579252100356
✦

¿Para qué sirve?



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

```
saved share run
```

```
main.py
1 import numpy
2
3 arreglo = numpy.random.random(10,)
4 mayor = numpy.max(arreglo)
5 print('El mayor elemento es', mayor)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
El mayor elemento es 0.8914038534619878
>
```


Arreglos



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)
```



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

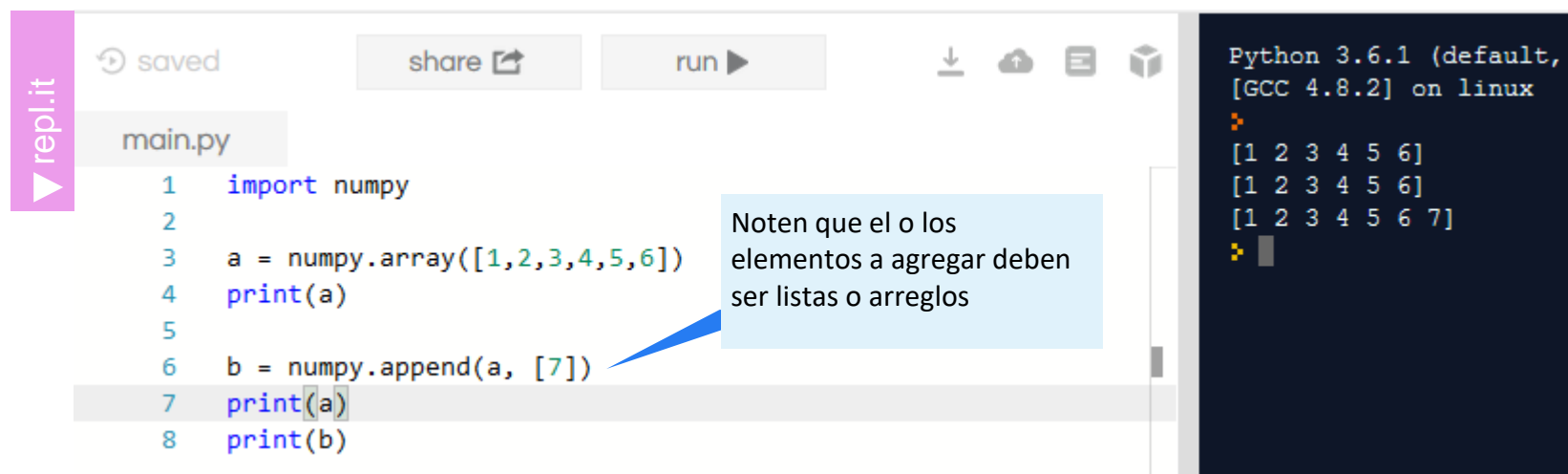
Ejemplo:

```
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



The screenshot shows a Jupyter Notebook interface. On the left, a pink sidebar contains the text 'repl.it'. The main area displays a Python script in a file named 'main.py'. The script is as follows:

```
1 import numpy
2
3 a = numpy.array([1,2,3,4,5,6])
4 print(a)
5
6 b = numpy.append(a, [7])
7 print(a)
8 print(b)
```

A blue callout box points to line 6 of the script, containing the text: 'Noten que el o los elementos a agregar deben ser listas o arreglos'. The right side of the interface shows the output of the script in a terminal window. The output is:

```
Python 3.6.1 (default,
[GCC 4.8.2] on linux
>
[1 2 3 4 5 6]
[1 2 3 4 5 6]
[1 2 3 4 5 6 7]
```

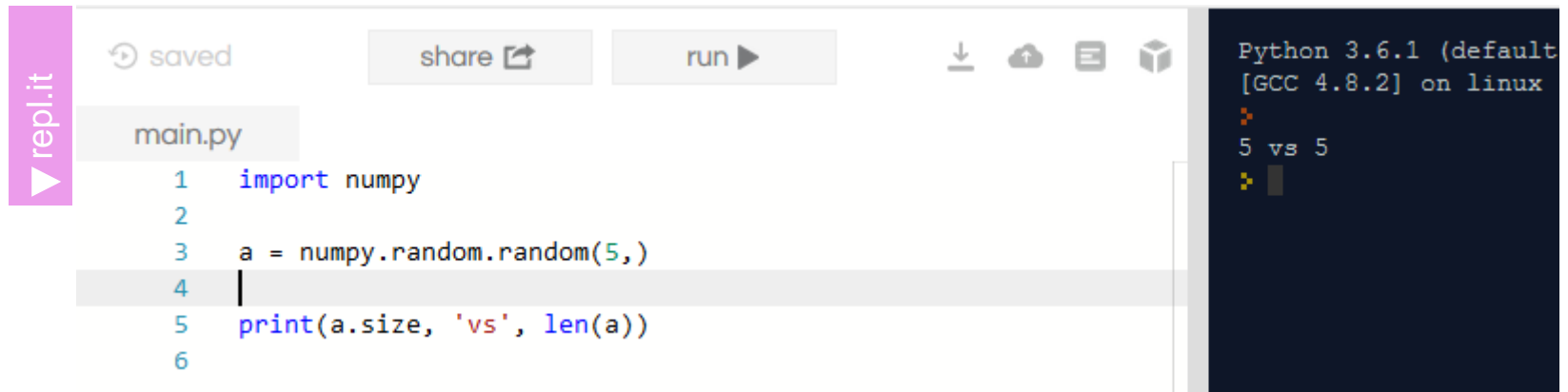
Importante

Los arreglos NO se actualizan cuando usamos `append`, sino que se crea una copia del arreglo con el 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`



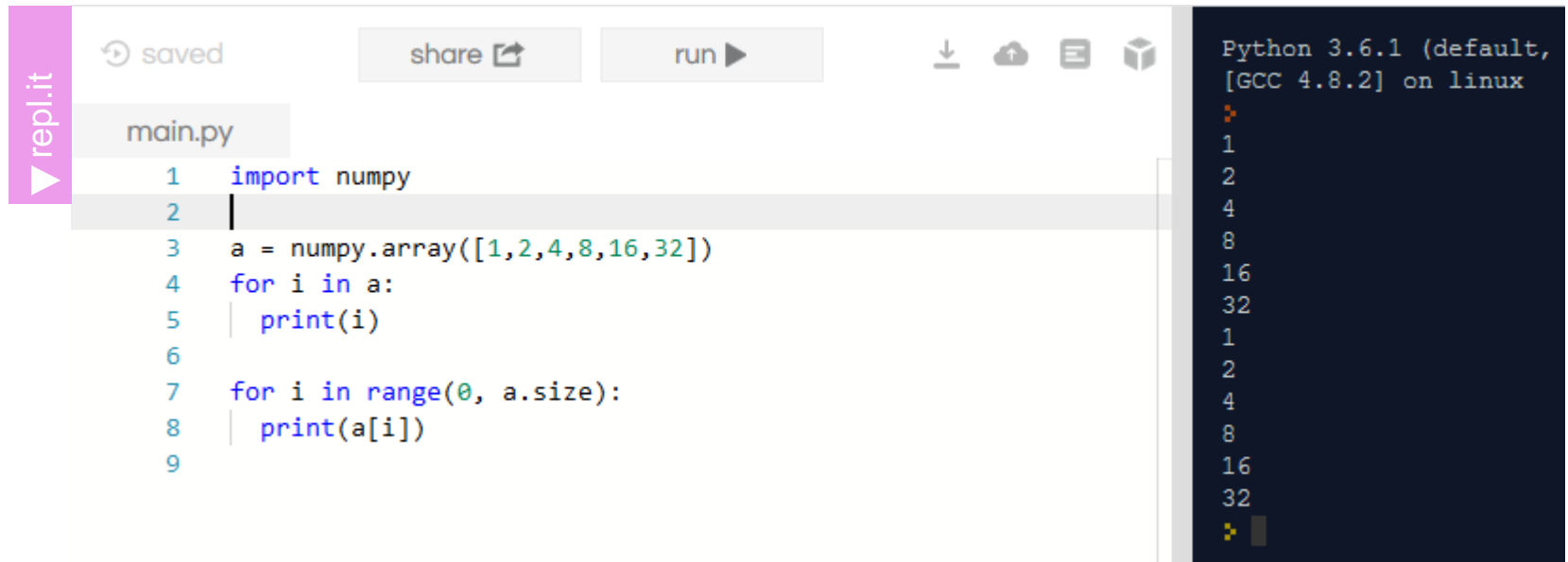
```
repl.it
saved share run
main.py
1 import numpy
2
3 a = numpy.random.random(5,)
4
5 print(a.size, 'vs', len(a))
6
```

Python 3.6.1 (default [GCC 4.8.2] on linux)
5 vs 5

Recorrer un arreglo



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



```
repl.it
saved share run
main.py
1 import numpy
2
3 a = numpy.array([1,2,4,8,16,32])
4 for i in a:
5     print(i)
6
7 for i in range(0, a.size):
8     print(a[i])
9

Python 3.6.1 (default,
[GCC 4.8.2] on linux
1
2
4
8
16
32
1
2
4
8
16
32
```

Sin embargo, lo realmente entretenido de numpy son los cálculos sobre los arreglos

- **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

Ejemplo

repl.it

saved

share

run



main.py

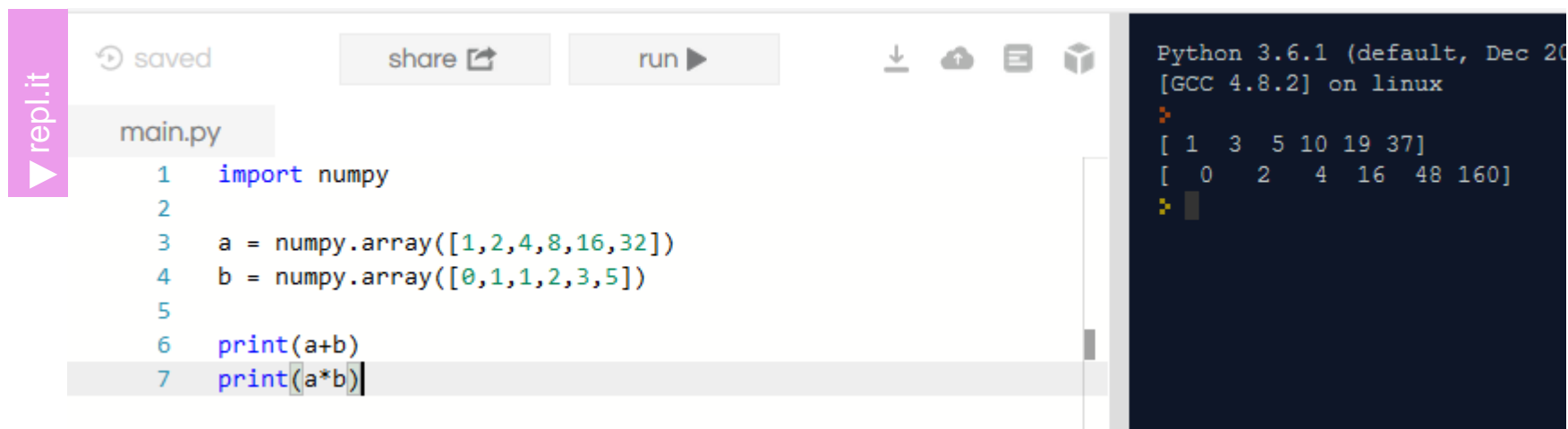
```
1 import numpy
2
3 a = numpy.array([1,2,4,8,16,32])
4 print(a * 4)
5
6 print('La suma de los elementos de a es', a.sum())
7 print(' y su multiplicación', a.prod())
8
9 print('El promedio de los elementos de a es', a.mean())
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
```

```
>
[ 4  8 16 32 64 128]
La suma de los elementos de a es 63
 y su multiplicación 32768
El promedio de los elementos de a es 10.5
>
```

Cálculos

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



The screenshot shows a Repl.it Python environment. On the left, a pink vertical bar contains the text 'repl.it'. The main editor area shows a file named 'main.py' with the following Python code:

```
1 import numpy
2
3 a = numpy.array([1,2,4,8,16,32])
4 b = numpy.array([0,1,1,2,3,5])
5
6 print(a+b)
7 print(a*b)
```

On the right, the terminal output shows the results of the code execution:

```
Python 3.6.1 (default, Dec 20
[GCC 4.8.2] on linux
[ 1  3  5 10 19 37]
[ 0  2  4 16 48 160]
```


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 evalúan como verdadero
- **arreglo.any():** retorna verdadero si alguno de los elementos de un arreglo se evalúan como verdadero

▼ repl.it

saved

share

run



main.py

```
1 import numpy
2
3 a = numpy.array([False, False, False, True, False])
4 b = numpy.array([1,1,1,1,1,1])
5
6 print(a.any())
7 print(a.all())
8
9 print(b.any())
10 print(b.all())
11
```

Python 3.6.1 (default
[GCC 4.8.2] on linux

```
>
True
False
True
True
>
```

Un científico norteamericano 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 (maldito sistema métrico inglés).

Escriba el código para transformar esas mediciones en centímetros

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

Ejemplo: Para el arreglo

```
notas = numpy.array([3.4, 4.3, 4.7, 5.1, 5.3, 5.8, 6.1, 6.7, 7.0])  
# acá va el resto del código
```

Debe mostrar

```
El promedio es 5.377777777777777  
La nota más baja es 3.4  
La nota más alta es 7.0
```

Solución

Un científico norteamericano 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 (maldito sistema métrico inglés).

Escriba el código para transformar esas mediciones en centímetros

```
import numpy

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

print(en_centimetros)
```

Solución

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

```
import numpy

notas = numpy.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)
```