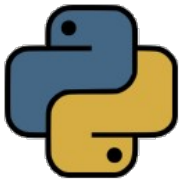
Pablo Alcain
pabloalcain@gmail.com

Introducción a la Programación en Python

¿Quién es python?

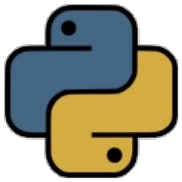
Es un lenguaje de programación interpretado, que permite tipado dinámico y es multiplataforma



¿Quién es python?

Es un lenguaje de programación interpretado, que permite tipado dinámico y es multiplataforma

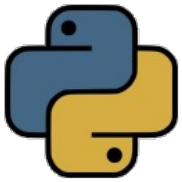
Multiparadigma:
- programación orientada a objetos
- programación imperativa
- programación funcional



¿Quién es python?

Es un lenguaje de programación **interpretado**, que permite tipado dinámico y es multiplataforma

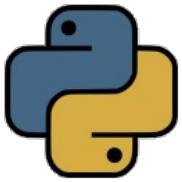
Multiparadigma:
- programación orientada a objetos
- programación imperativa
- programación funcional



¿Quién es python?

Es un lenguaje de programación **interpretado**, que permite tipado **dinámico** y es multiplataforma

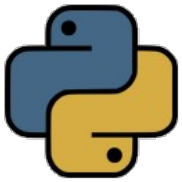
Multiparadigma:
- programación orientada a objetos
- programación imperativa
- programación funcional



¿Quién es python?

Es un lenguaje de programación **interpretado**, que permite tipado **dinámico** y es **multiplataforma**

Multiparadigma:
- programación orientada a objetos
- programación imperativa
- programación funcional



¿Por qué python...

... si yo ya sé programar en *?

Es **fácil** de aprender

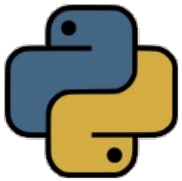
Gran conjunto de **librerías**

Excelente soporte **científico**

El software se puede desarrollar bastante **rápido** y relativamente **compacto**

Licencia de **código abierto** (en tu cara MATLAB)

Comunidad muy amplia y consciente del uso científico



OK, pero... ¿por qué programar?

Específicamente la **Física Computacional** sigue creciendo

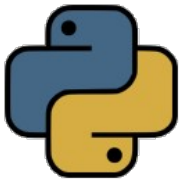
Uno nunca sabe cuándo puede **necesitar** una computadora

... o **comunicarse** con alguien que sepa programar

Analizar **datos** experimentales

Sacarle el jugo a la computadora

Porque viene una materia y **¡zas!**, les obliga



El intérprete de python

Un modo interactivo (aprovechando que es interpretado)

Introducir una a una las expresiones

Probar porciones de código

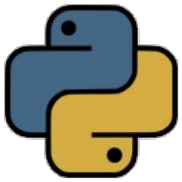
```
$ python
>>> 1 + 1
2
>>> a = 1 + 1
>>> print a
2
>>> a = range(10)
>>> print a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

prompt del intérprete de python

asignación de variable

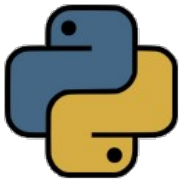
imprime en pantalla

tipado dinámico



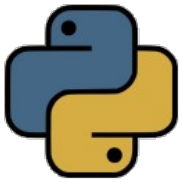
Algunos tipos de python

```
>>> type('hello, world!')
<type 'str'>
```



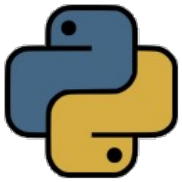
Algunos tipos de python

```
>>> type('hello, world!')
<type 'str'>
>>> type(1)
<type 'int'>
```



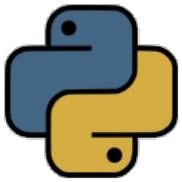
Algunos tipos de python

```
>>> type('hello, world!')
<type 'str'>
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
```



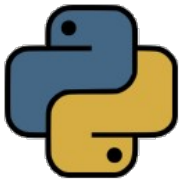
Algunos tipos de python

```
>>> type('hello, world!')
<type 'str'>
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
>>> type([1, 2, 3, 4])
<type 'list'>
```



Algunos tipos de python

```
>>> type('hello, world!')
<type 'str'>
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
>>> type([1, 2, 3, 4])
<type 'list'>
>>> type((1, 2, 3, 4))
<type 'tuple'>
```

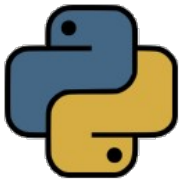


Detalles de la sintaxis

¡El espacio en blanco importa!

```c
/* file: if.c */
if (a > b) {
  printf("a is greater\n");
  printf("b is less\n");
}
```

```python
# file: if.py
if a > b:
  print "a is greater"
  print "b is less"
```



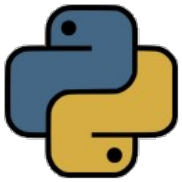
Detalles de la sintaxis

¡El espacio en blanco importa!

```c
/* file: if.c */
if (a > b) {
  printf("a is greater\n");
  printf("b is less\n");
}
```

```python
# file: if.py
if a > b:
  print "a is greater"
  print "b is less"
```

```
>>> from future import braces
```



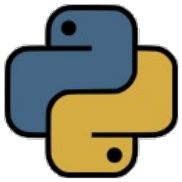
Detalles de la sintaxis

¡El espacio en blanco importa!

```c
/* file: if.c */
if (a > b) {
  printf("a is greater\n");
  printf("b is less\n");
}
```

```python
# file: if.py
if a > b:
  print "a is greater"
  print "b is less"
```

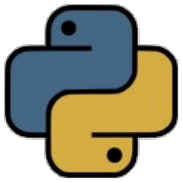
```
>>> from future import braces
  File "<stdin>", line 1
SyntaxError: not a chance
```



Control de flujo

```
# file: for.py
for name in ["Jorge", "Claudia", "Pablo"]:
  print name
```

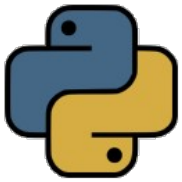
```
$ python for.py
```



Control de flujo

```
# file: for.py
for name in ["Jorge", "Claudia", "Pablo"]:
  print name
```

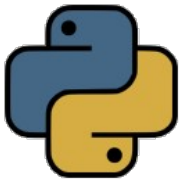
```
$ python for.py
Jorge
Claudia
Pablo
```



Control de flujo

```python
# file: if_elif.py
a = 1
b = 2
if a > b:
  print "a is greater than b"
elif a > 0:
  print "a is not greater than b, but greater than 0"
else:
  print "a is less than b and less than 0"
```

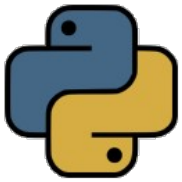
```
$ python if_elif.py
```



Control de flujo

```
# file: if_elif.py
a = 1
b = 2
if a > b:
  print "a is greater than b"
elif a > 0:
  print "a is not greater than b, but greater than 0"
else:
  print "a is less than b and less than 0"
```

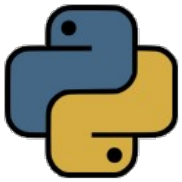
```
$ python if_elif.py
a is not greater than b, but greater than 0
```



Control de flujo

```
# file: while.py
a = 5
while a > 2:
  print a
  a = a - 1
```

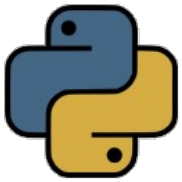
```
$ python while.py
```



Control de flujo

```python
# file: while.py
a = 5
while a > 2:
  print a
  a = a - 1
```

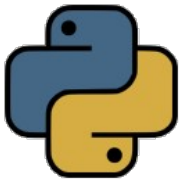
```
$ python while.py
5
4
3
```



Control de flujo

```python
# file: break.py
for i in range(5):
  if i == 3:
    break
  print i
```

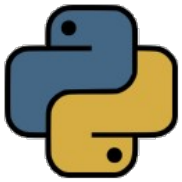
```
$ python break.py
```



Control de flujo

```python
# file: break.py
for i in range(5):
  if i == 3:
    break
  print i
```

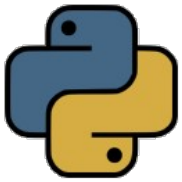
```
$ python break.py
0
1
2
```



Control de flujo

```python
# file: continue.py
for i in range(5):
  if i == 3:
    continue
  print i
```

```
$ python continue.py
```



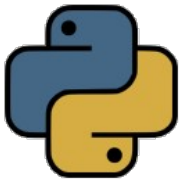
Un poco de strings...

```
>>> s = 'hello, world!'
>>> print s
hello, world!
>>> s = '''hola a todos
... como les va?'''
>>> print s
hola a todos
como les va?
>>> s = 'Me llamo {} y mi edad es {}.'
>>> print s.format('Mafalda', 6)
Me llamo Mafalda y mi edad es 6
>>> s = 'Me llamo {nombre} y mi edad es {edad}'
>>> print s.format(edad=6, nombre='Mafalda')
Me llamo Mafalda y mi edad es 6
```

Puedo usar ' o " indistintamente

Triple comillas ''': verbatim quotes

Formato con o sin variables



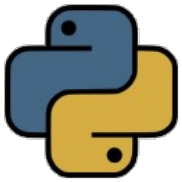
Algunas cosillas

en python2 la división es **entera**

```
>>> 5/3
1
>>> 5.0/3.0
1.666666666667
```

Los operadores booleanos son explícitos:

```
#file: boolean.py
True
False
True and False
False or True
not True
```



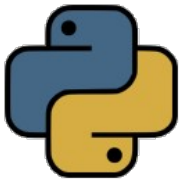
Definición de funciones

```c
/* file: factorial.c */
int factorial(int n) {
  if (n == 0) {
    return 1;
  }
  else {
    return n * factorial(n - 1);
  }
}
```

La indentación es opcional, los bloques los definen las llaves

```python
# file: factorial.py
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

La indentación es obligatoria, define los bloques



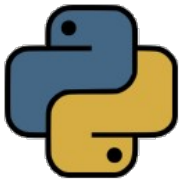
Librerías Científicas

¿El verdadero motivo por el que usamos python?

NumPy: `http://www.numpy.org`

SciPy: `http://www.scipy.org`

MatPlotLib: `http://www.matplotib.org`



NumPy

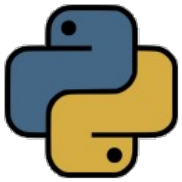
Fundamental para la programación científica en python

Agrega un tipo fundamental: el **NumPy array**

Potentes arrays N-dimensionales

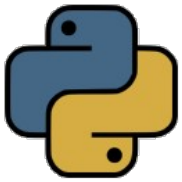
Herramientas para integración con código de C y FORTRAN

Álgebra lineal, transformadas de Fourier, números aleatorios...



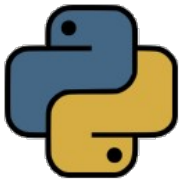
El array de numpy

```
>>> import numpy as np
>>> a = np.array([1, 4, 5, 8])
>>> a
array([1, 4, 5, 8])
>>> type(a)
<type 'numpy.ndarray'>
>>> a[2]
5
>>> a[1:]
array([4, 5, 8])
```



El array de numpy

```
>>> import numpy as np
>>> a = np.linspace(0, 1, 5)
>>> print a
[ 0.     0.25   0.5    0.75   1.  ]
>>> b = np.linspace(0, 2, 5)
>>> print b
[ 0.    0.5   1.    1.5   2. ]
>>> print a + b
[ 0.    0.75    1.5   2.25    3.  ]
```

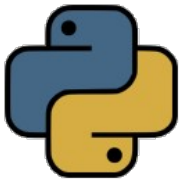


MatPlotLib

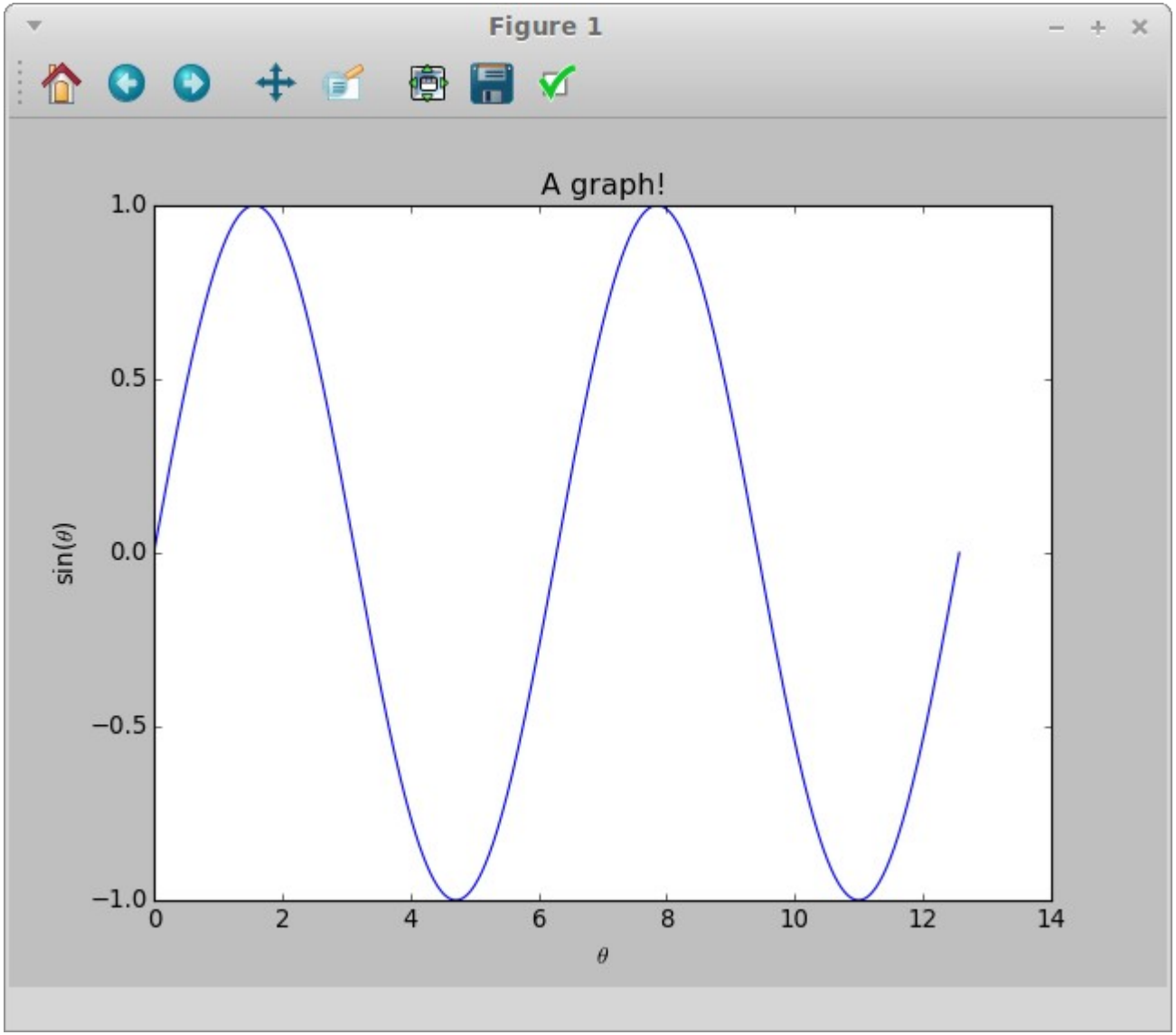
Librería para producir gráficos de alta calidad programáticamente

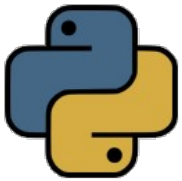
Generalmente llamamos al combo NumPy + MatPlotLib = PyLab

```
>>> import pylab as pl
>>> x = pl.linspace(0, 4 * pl.pi, 1000)
>>> y = pl.sin(x)
>>> pl.plot(x, y, label='sin')
>>> pl.xlabel(r'$\theta$')
>>> pl.ylabel(r'sin($\theta$)')
>>> pl.title('A graph!')
>>> pl.show()
```



MatPlotLib





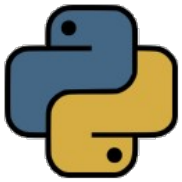
¿Qué les damos?

Un programa en python ya escrito que calcula las funciones de onda y las energías de estados ligados y libres.

Básicamente está todo el trabajo hecho. Les queda generar los resultados y graficarlos, etcétera.

Para eso pusimos un script de python que corre algunas situaciones típicas y hace algunos gráficos

https://github.com/pabloalcain/hydrogenic



Pablo Alcain
pabloalcain@gmail.com

Introducción a la Programación en Python