

# Introducción a Python

Edgar Acuna

[edgar.acuna@upr.edu](mailto:edgar.acuna@upr.edu)

[website:academic.uprm.edu/eacuna](http://academic.uprm.edu/eacuna)

Agosto del 2017

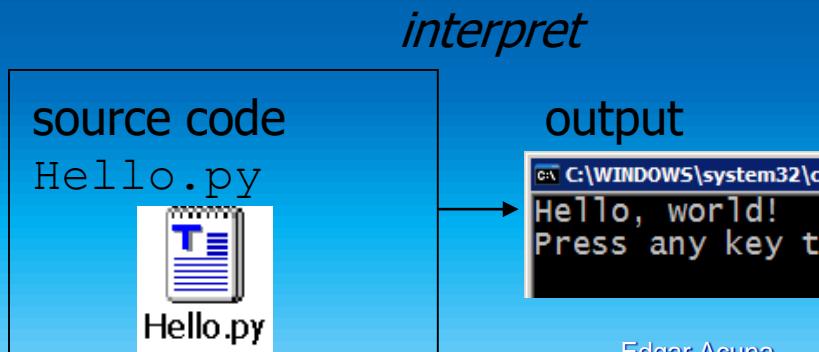
Edgar Acuna

# Compilacion e interpretacion

- Muchos lenguajes requieren compilacion (traduccion) del programa a una forma que la computadora entienda.



Python (al igual que R) es directamente *interpretado* en instrucciones de computadora.



# 4 versiones principales de Python

- “Python” or “CPython” is written in C/C++
  - Version 2.7 14. Septiembre 16, 2017
  - Version 3.6.4 Diciembre, 17 2017
- “Jython” esta escrita en Java. La ultima version salio en Mayo 2015.
- “IronPython” esta escrita en C# para el .Net entorno (version 2.7.7 salio en diciembre 2016).

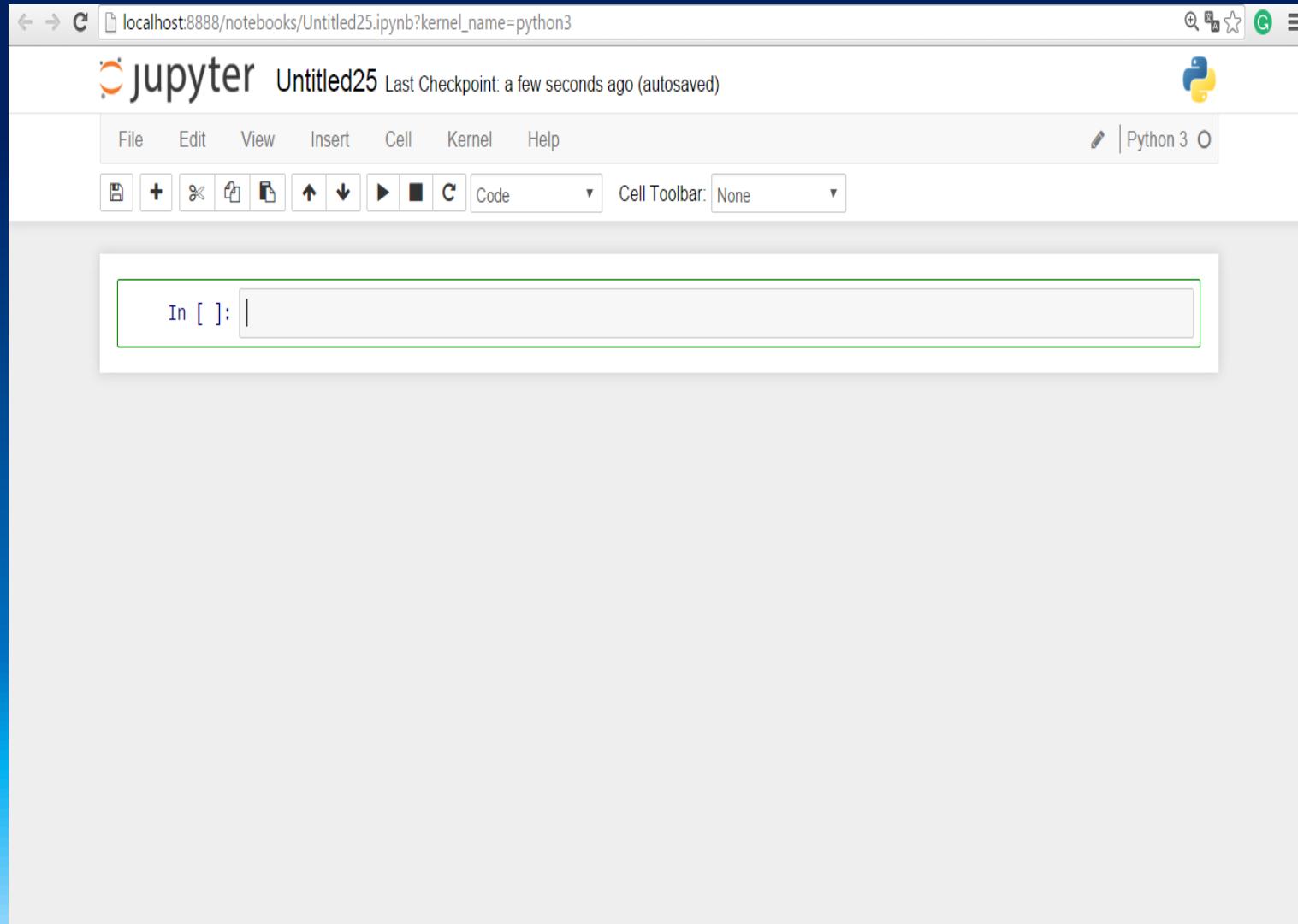
# Instalando Python en Windows

1. Descargar de Anaconda ([www.continuum.io](http://www.continuum.io)) ambas versions de python 2.7 y 3.6.
- 2- Instalar python3 siguiendo las instrucciones disponibles en <https://docs.continuum.io/anaconda/install/windows>  
Usar su equivalente en MAC y Linux.
- 3- Dar el comando `jupyter notebook` para iniciar lo usando el kernel de python 3
- 3- Para añadir el kernel python 2 al jupyter notebook, ejecutar los siguientes comandos:  
`conda create -n ipykernel_py2 python=2 ipykernel`  
`activate ipykernel`  
`python -m ipykernel install --user`

# Ambientes de Desarrollo (IDE)

1. Emacs (Linux)
2. Vim(Linux)
3. Idle (Windows)
4. NotePad++ (Windows)
5. Spyder
6. Jupyter Notebook ( Windows, Linux, MacOS)

# Jupyter Notebook



# Python Interactive Shell

```
% python
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:15:05)
[MSC v.1600 32 bit<intel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Se puede correr cosas directamente en una session de Python

```
>>> 2+3*4
14
>>> name = "Andrew"
>>> name
'Andrew'
>>> print ("Hello", name)
Hello Andrew
>>>
```

# Ipython= Un Python Interactive Shell mejorado

```
% ipython
Python 2.7.9 | Anaconda 2.30 64 bits | default Dec 18 2014, 16:57:52
[MSC v.1500 64 bit<AMD 64>]
Type "help", "copyright", "credits" or "license" for more information.
```

Ipython 3.2.1. An enhanced interactive Python.  
Anaconda is brought to you by Continuum Analytics.  
Please check out <http://continuum.io/thanks> and <https://anaconda.org>

?            -> Introduction and overview of Ipython's features..
%quickref -> Quick Reference.
help        -> Python's own help system.
Object      -> Details about 'object', use 'object??' for further details.

In [1]:

# Expresiones

- **Expresión:** Es el valor de un dato o un conjunto de operaciones para calcular un valor.

## Ejemplos:

$$1 + 4 * 3$$

13

- Operadores aritmeticos:

- `+ - * /` suma, resta, multiplicacion, division
  - `%` module (o residuo)
  - `**` exponenciacion

# Logica

- Muchas expresiones logicas usan operadores relacionales:

Operador	Significado	Ejemplo	Resultado
<code>==</code>	equals	<code>1 + 1 == 2</code>	True
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	True
<code>&lt;</code>	less than	<code>10 &lt; 5</code>	False
<code>&gt;</code>	greater than	<code>10 &gt; 5</code>	True
<code>&lt;=</code>	less than or equal to	<code>126 &lt;= 100</code>	False
<code>&gt;=</code>	greater than or equal to	<code>5.0 &gt;= 5.0</code>	True

Operador	Ejemplo	Resultado
<code>and</code>	<code>9 != 6 and 2 &lt; 3</code>	True
<code>or</code>	<code>2 == 3 or -1 &lt; 5</code>	True
<code>not</code>	<code>not 7 &gt; 0</code>	False

# input

- input :Lee un numero entrado por el usuario.
  - Se puede asignar (almacenar) el resultado de input a una variable.
  - Ejemplo(py2)

```
age = input("How old are you? ")  
print "Your age is", age  
print "You have", 65 - age, "years  
until retirement"
```

## Output:

How old are you? 53

Your age is 53

You have 12 years until retirement

print

- print : muestra un mensaje de texto o el valor de una expresion en la pantalla.
- Syntax:

print "Message"

print Expression

**Nota: En Python 3.4 se usa parentesis pero en la 2.7 no.**

print (*Item1*, *Item2*, ..., *ItemN*)

- Imprime varios mensajes y/o expresiones en una misma linea.

# Strings

- **string:** Una secuencia de caracteres textuales en un programa.

- Strings empiezan y terminan en un character " o apostrofe ' .
- Ejemplos:

"hello"

"This is a string"

"This, too, is a string. It can be very long!"

- Un string no puede escribirse en varias lineas ni contener un caracter ".

"This is not  
a legal String."

"This is not a "legal" String either."

- Un string puede representar caracteres precediendoles con un backslash.

- \t tab character
- \n new line character
- \" quotation mark character
- \\ backslash character

- Example: "Hello\tthere\nHow are you?"

# Propiedades de Strings

- `len(string)` - numero de caracteres en un string (incluyendo espacios en blanco)
- `str.lower(string)` - lowercase version of a string
- `str.upper(string)` - uppercase version of a string
- Ejemplo:

```
name = "Edgar Acuna Fernandez"
length = len(name)
big_name = str.upper(name)
print big_name, "tiene", length,
"caracteres"
```

# Listas

Es un objeto que puede contener distintos tipos de datos:

A=[0]

B=[2.3, 4.5]

C=[5, "Hello", "there", 9.8]

D=[]

Usar `len()` para obtener la longitud de una lista

```
>>> names = ["Ana", "Rosa", "Julia"]
```

```
>>> len(names)
```

3

# Uso de [ ] para indexar items en una lista

```
>>> names[0]  
'Ana'  
>>> names[1]  
'Rosa'  
>>> names[2]  
'Julia'  
>>> names[3]  
Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
IndexError: list index out of range  
>>> names[-1]  
'Julia'  
>>> names[-2]  
'Rosa'  
>>> names[-3]  
'Ana'
```

[0] es el primer item.  
[1] es el segundo item

...

Valor Out of range da un exception

Valores negativos van hacia atras comenzando desde el ultimo elemento

# Range()

- La function “range” crea una lista de numeros enteros en un rango especificado
- range([start], [stop], [step]) -> lista de enteros
- El valor de step especifica el incremento o decremento de la secuencia.

```
>>> range(5)
[0, 1, 2, 3, 4]
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 2)
[0, 2, 4, 6, 8]
```

La funcion range funciona en forma distinta en la version 3 de Python

# if

- **Enunciado if :** Ejecuta un grupo de enunciados solo si una condición dada es cierta. De lo contrario los enunciados son omitidos.

- Syntax:

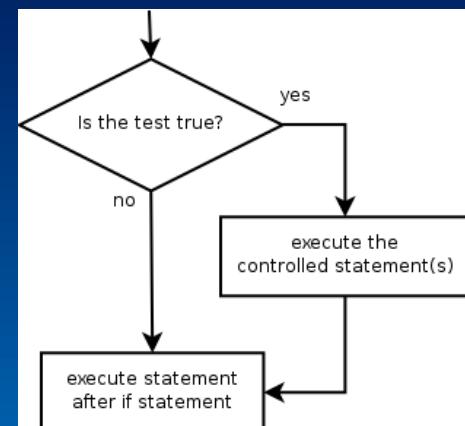
```
if condicion:  
    enunciados
```

- Ejemplo:

```
gpa = 3.4
```

```
if gpa > 2.0:
```

```
    print "Your application is  
accepted."
```



# if/else

- **Enunciado if/else.** Ejecuta un bloque de enunciados si una cierta condicion se cumple, y un segundo bloque de enunciados si la condicion no se cumple .

- Syntax:

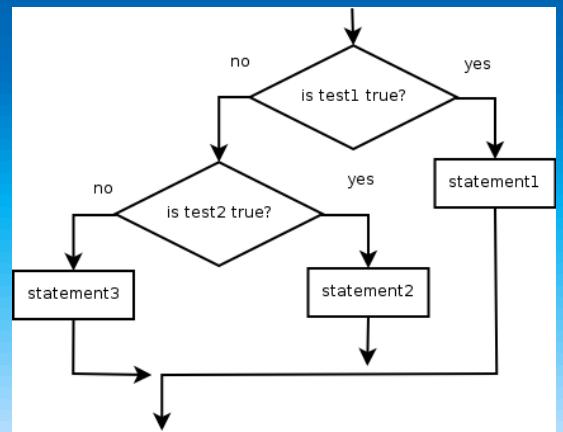
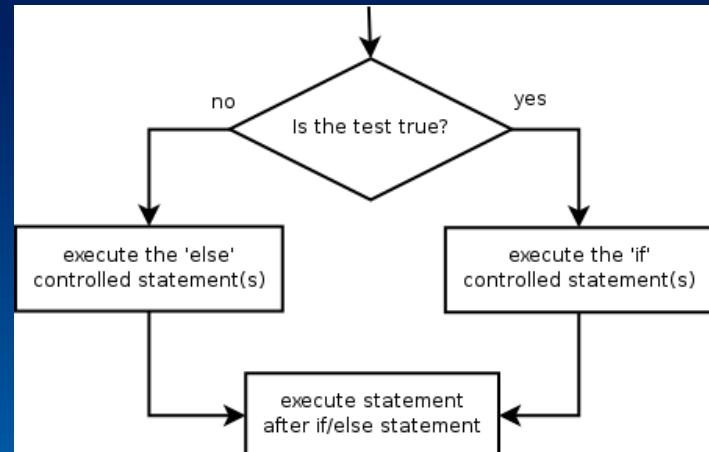
```
if condition:  
    statements  
else:  
    statements
```

- Ejemplo:

```
gpa = 1.4  
if gpa >= 2.5:  
    print "Welcome to Mars University!"  
else:  
    print "Your application is denied."
```

- Si hay Multiple condiciones se usa elif ("else if"):

```
if condition:  
    statements  
elif condition:  
    statements  
else:
```



# Logica Booleana

Las expresiones en Python pueden tener “and”s y “or”s:

Ana=3

Rosa=25

if (Ana <= 5 and Rosa >= 10 or

Rosa == 500 and Ana != 5):

print “Ana and Rosa”

# El loop for [1]

- **El loop for:** Repite un conjunto de enunciados para un conjunto de valores.

- Syntax:

```
for variableName in groupOfValues:  
    statements
```

- Los enunciados a ser repetidos son indentados con tabs or spaces.
- **variableName** da un nombre a cada valor, paara que pueden ser feferidos en los enunicados
- **groupOfValues** puede ser un rango de enteros especificados con la funcion ,range .

- Ejemplo:

```
for x in range(1, 4):  
    print x, "squared is", x * x
```

Output:

```
1 squared is 1  
2 squared is 4  
3 squared is 9
```

# El loop for [2]

```
>>> names = ["Ana", "Rosa", "Julia"]
```

```
>>> for name in names:
```

```
...[redacted]print name
```

```
...
```

Ana

Rosa

Julia

# Break, continue

```
>>> for value in [3, 1, 4, 1, 5, 9, 2]:  
...     print "Checking", value  
...     if value > 8:  
...         print "Exiting for loop"  
...         break  
...     elif value < 3:  
...         print "Ignoring"  
...         continue  
...     print "The square is", value**2
```

Checking 3  
The square is 9  
Checking 1  
Ignoring  
Checking 4  
The square is 16  
Checking 1  
Ignoring  
Checking 5  
The square is 25  
Checking 9  
Exiting for loop  
>>>

# while

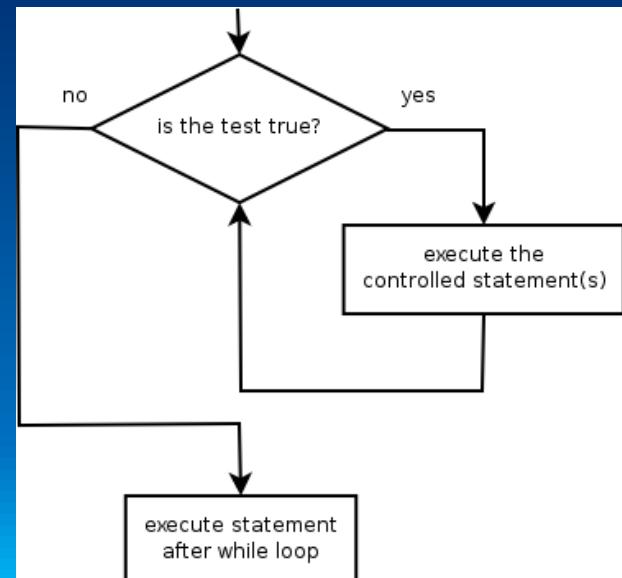
- El loop while; Ejecuta iterativamente un grupo de enunciados siempre que una condicion dada sea cierta.
  - Bueno para loops *indefinidos* (repetir un enunciado un numero desconocido de veces)
- Syntax:

**while *condicion*:**  
***enunciado***

- Example:

number = 1

**while number < 200:**  
    print number,  
    number = number \* 2



# Procesamiento de Files

- Muchos programas manipulan datos, que a menudo estan en files.
- Para leer el contenido de un file en python 2 usar:

```
variableName =  
    open ("filename").read()
```

Example:

```
file_text =  
    open ("c://esma3016/Animals2.csv").  
    read()
```

# Procesamiento Linea-por-linea

- Leyendo un file line-by-line:

```
for line in open("filename").readlines():  
    statements
```

Ejemplo: Imprime linea por linea un file

```
count = 0  
for line in  
    open("c://esma3016/Animals2.csv").readlines():  
        print line  
        count = count + 1  
print "El archivo contiene", count, "lineas."
```

# Una forma mas rapida de leer

```
>>> lst= [ x for x in open("c://esma3016/Animals2.csv","r").readlines() ]
```

```
>>> lst
```

Para ignorar el header usar:count = 0

```
for line in open("c://esma3016/Animals2.csv").readlines():
```

```
    if count!=0:
```

```
        print line
```

```
    count = count + 1
```

```
print "The file contains", count, "lineas."
```

Tambien se puede leer datos directamente de la web usando el modulo urllib, pero la forma mas facil de leer un file es usando el modulo pandas (ver slide mas Adelante)

# File Output

```
input_file = open("in.txt")
output_file = open("out.txt", "w")
for line in input_file:
    output_file.write(line)
```

- 
- “w” = “write mode”
  - “a” = “append mode”
  - “wb” = “write in binary”
  - “r” = “read mode” (default)
  - “rb” = “read in binary”
  - “U” = “read files with Unix or Windows line endings”

# Modulos

- Un programa en Python inicialmente solo tiene acceso a unas funciones basicas.  
("int", "dict", "len", "sum", "range", ...)
- `dir(__builtins__)` da una lista de las funciones disponibles.
- El uso de "Modulos" le anade mas funcionalidad a Python. Para cargar un modulo se usa el comando "import".
- Ejemplos

```
>>> import math #Calculo de funciones matematicas  
>>> import Numpy #Contiene calculos y graficas estadisticos.
```

# Mas modulos

```
>>>import scipy # Para hacer calculos científicos tales como  
    integracion numerica y optimizacion  
>>> import matplotlib # Para hacer graficas al estilo de matlab  
>>> import pandas #Para hacer analisis estadistico  
>>> import statsmodels #Para hacer regresion y series de tiempos
```

# Usando el modulo math

```
>>> import math
>>> math.pi
3.1415926535897931
>>> math.cos(0)
1.0
>>> math.cos(math.pi)
-1.0
>>> dir(math)
['__doc__', '__file__', '__name__', '__package__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'exp', 'fabs', 'factorial', 'floor', 'fmod',
'frexp', 'fsum', 'hypot', 'isinf', 'isnan', 'ldexp', 'log', 'log10',
'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'trunc']
>>> help(math)
>>> help(math.cos)
```

# Comandos del modulo math

- Python tiene varios comandos para hacer calculos.

Nombre del comando	Descripcion
abs ( <b>value</b> )	absolute value
ceil ( <b>value</b> )	rounds up
cos ( <b>value</b> )	cosine, in radians
floor ( <b>value</b> )	rounds down
log ( <b>value</b> )	logarithm, base e
log10 ( <b>value</b> )	logarithm, base 10
max ( <b>value1</b> , <b>value2</b> )	larger of two values
min ( <b>value1</b> , <b>value2</b> )	smaller of two values
round ( <b>value</b> )	nearest whole number
sin ( <b>value</b> )	sine, in radians
32	Edgar Acuna
sqrt ( <b>value</b> )	square root

Constante	Descripcion
e	2.7182818...
pi	3.1415926...

# “import” y “from ... import ...”

```
>>> import math as m
```

```
m.cos
```

```
>>> from math import cos, pi
```

```
cos
```

```
>>> from math import *
```

# Leyendo datos con Pandas

```
# desde un file en su propia computadora  
train=pd.read_csv('c:/Users/edgar2017/Downloads/titanic.csv')  
#directamente de la internet  
#na_values representa los valores faltantes(missing values)  
breastdf=pd.read_csv("https://archive.ics.uci.edu/ml/machine-  
learning-databases/breast-cancer-wisconsin/breast-cancer-  
wisconsin.data",header=None, sep=",",na_values=['?'])
```

# Referencias

<http://python.org/>

- documentacion, tutoriales, guias para principantes, distribucion core, ...

Libros:

- *Learning Python (2013)* by Mark Lutz (disponible en Piazza)
- *Python Essential Reference (2009)* by David Beazley  
(online at <http://code.activestate.com/recipes/langs/python/>)
- <http://wiki.python.org/moin/PythonBooks>
- Google's Python  
<https://developers.google.com/edu/python/>
- Dive into Python. <http://www.diveintopython.net/>