

Introduction to Python

Edgar Acuna

edgar.acuna@upr.edu

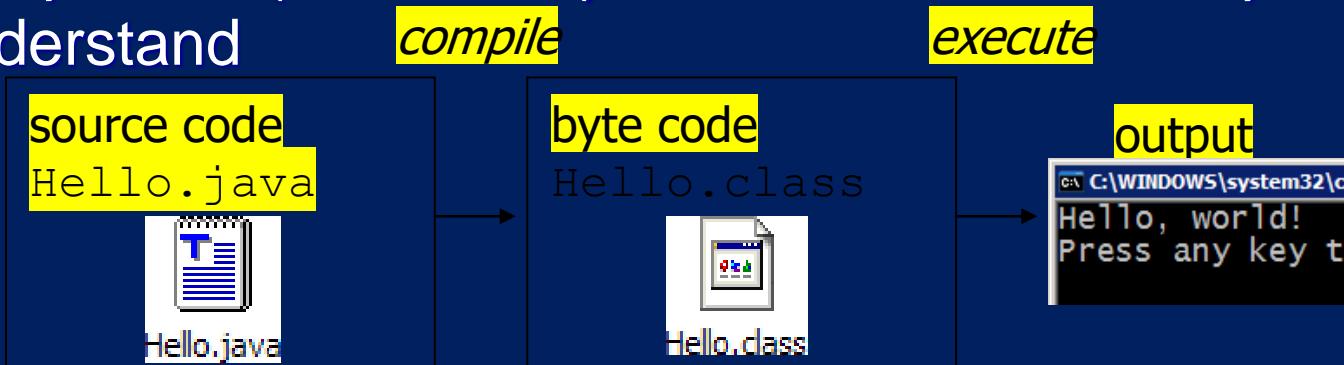
[website:academic.uprm.edu/eacuna](http://academic.uprm.edu/eacuna)

[Github:github.com/eacunafer](https://github.com/eacunafer)

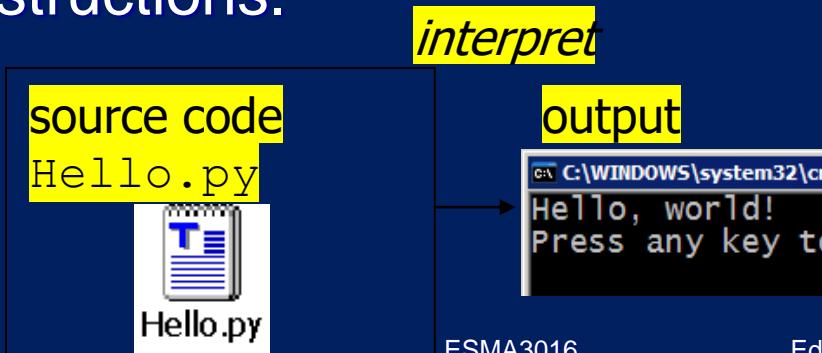
Agosto 15, 2020

Compilation and interpretation

- Many programming languages require program compilation (translation) to a form that the computer can understand



Python (same as R) is directly *interpreted in computer instructions*.



Four versions of Python

- “Python” or “CPython” is written in C/C++
 - Version 2.7.18. April 20, 2020
 - Version 3.7.7. March 10, 2020
- “Jython” is written in Java. The last version 2.7.2 was released in March 2020.
- “IronPython” is written in C# for the .Net environment (version 2.7.10 was released in Abril 27, 2020).

Installing Python in Windows

1. Descargar de Anaconda (www.anaconda.org) ambas versions de python 2.7 y 3.7.
- 2- Instalar python3 siguiendo las instrucciones disponibles en <https://www.anaconda.com/distribution/#download-section> Usar su equivalente en MAC y Linux.
- 3- Dar el comando jupyter notebook para iniciar lo usando el kernel de python 3

- 4- Para anadir 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
```

Aunque mejor es hacer la instalacion usando Anaconda Navigator.

Integrated Development environment (IDE) for Python

1. Emacs (Linux)
2. Vim(Linux)
3. Idle (Windows)
4. NotePad++ (Windows)
5. Spyder
6. Jupyter Notebook (Windows, Linux, MacOS)
7. Jupyter Lab (Febrero, 2018)

Python Interactive Shell

En Anaconda Navigator escoger Environment lego base(root) y luego open Terminal

C:\Users\leacun\Anaconda3>python

Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:

This Python interpreter is in a conda environment, but the environment hasnot been activated. Libraries may fail to load. To activate this environment please see <https://conda.io/activation>

Type "help", "copyright", "credits" or "license" for more information.

One can run python statements in the shell

```
>>> 2+3*4
```

```
14
```

```
>>> name = "Camila"
```

```
>>> name
```

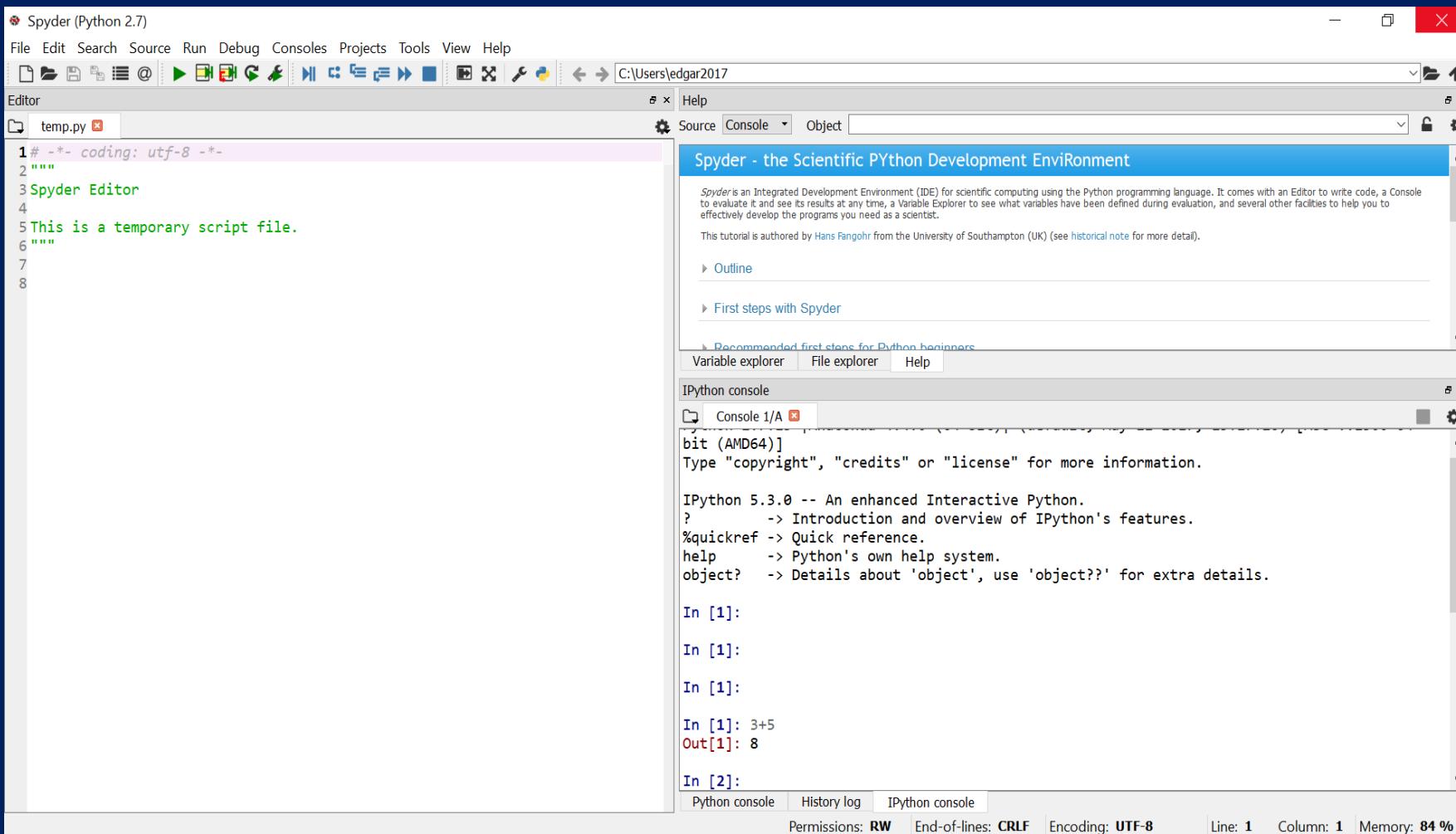
```
'Camila'
```

```
>>> print ("Hola", name)
```

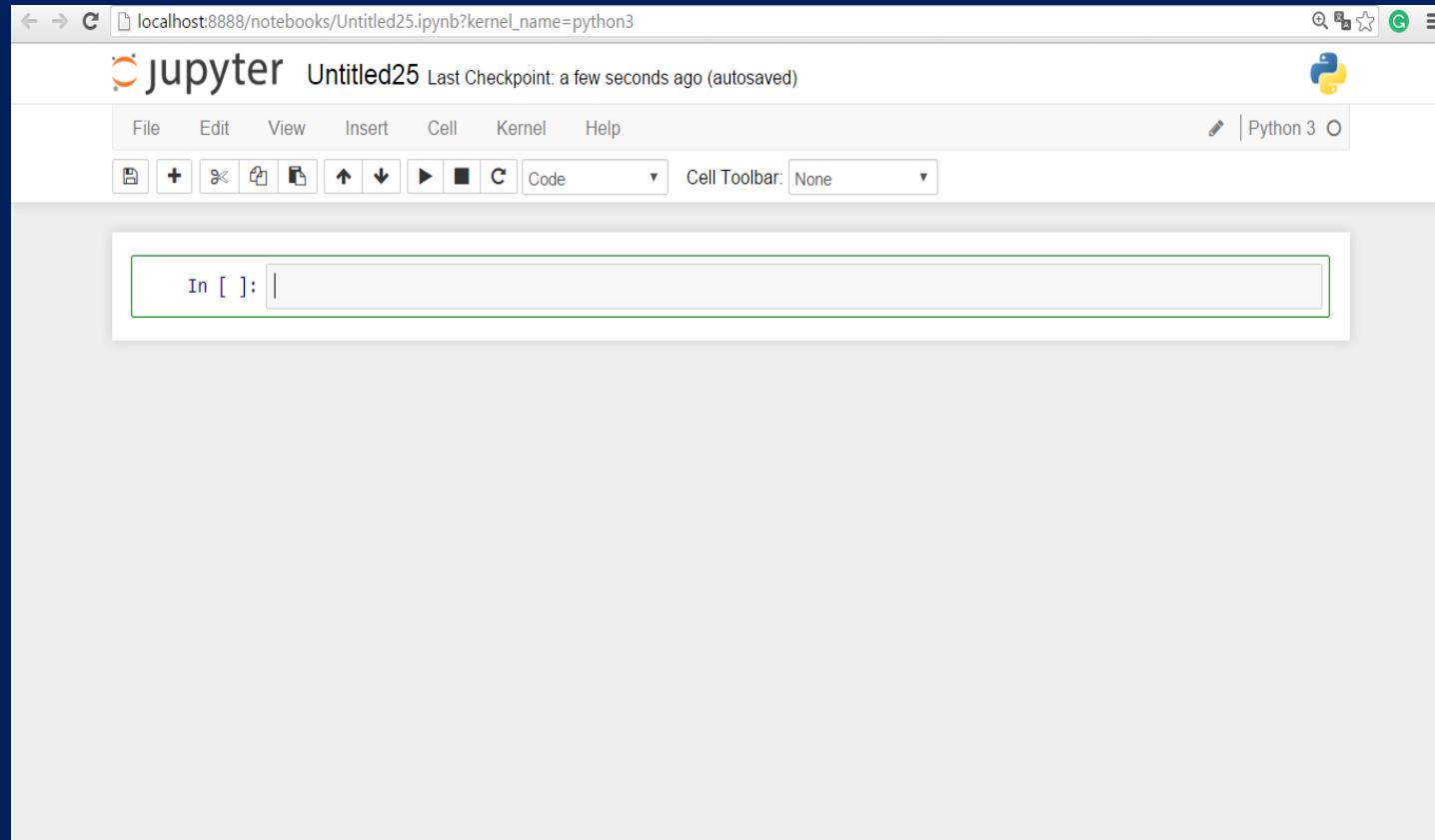
```
Hola Camila
```

```
>>>exit()
```

Spyder

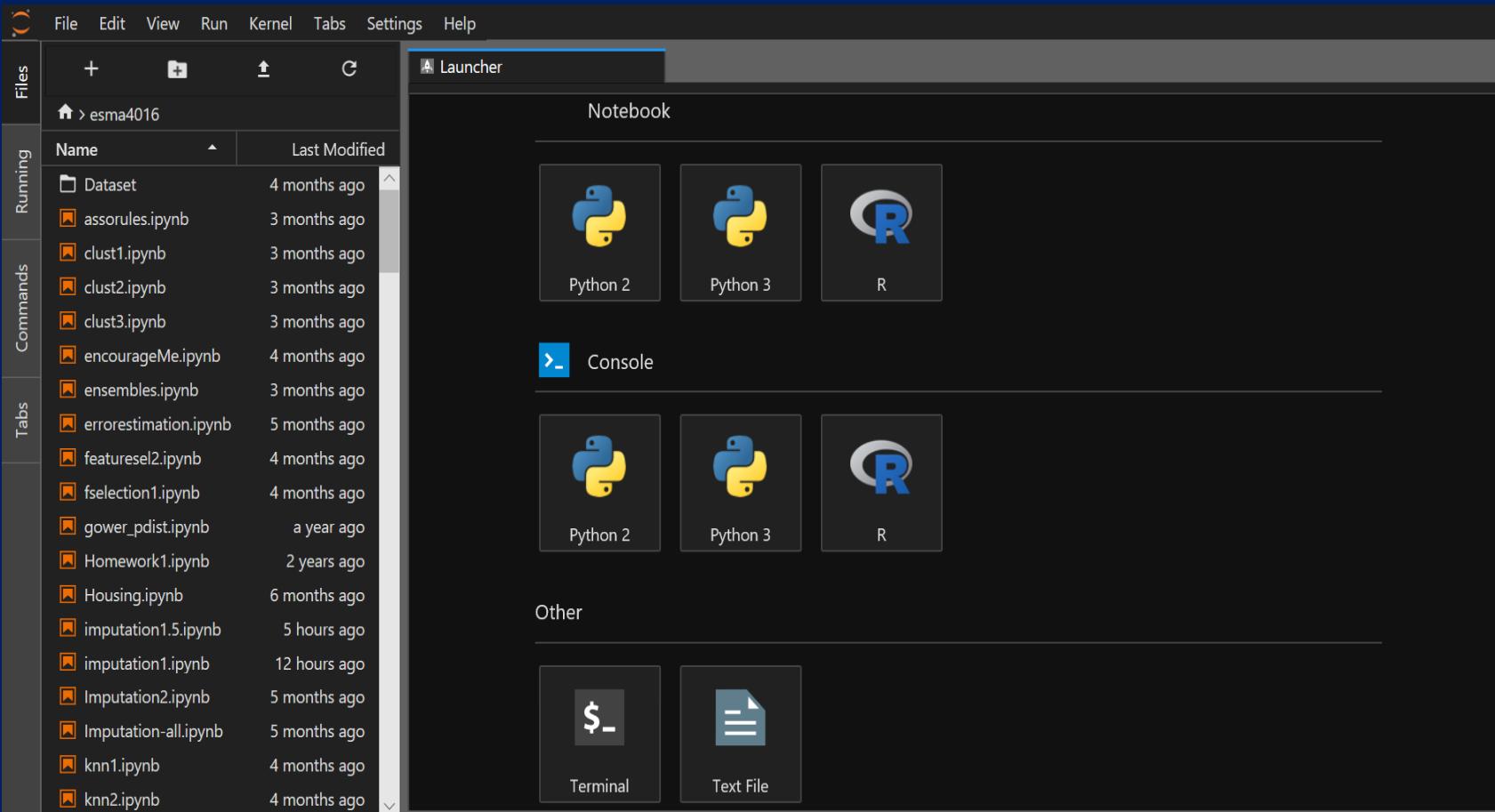


Jupyter Notebook



>Jupyter notebook

Jupyter Lab



>jupyter lab

Expressions

- **Expression:** It is a data value or un set of operations to compute a value.

Examples: $1 + 4 * 3$

13

- Arithmetic Operators:

- + - * / addition, substraction, multiplication, division
- % module (o residue)
- ** exponentiation

Logical Operators

- Muchas expresiones usan operadores logicos

Operador	Significado	Ejemplo	Resultado
<code>==</code>	Igual	<code>1 + 1 == 2</code>	True
<code>!=</code>	No es igual	<code>3.2 != 2.5</code>	True
<code><</code>	Menor que	<code>10 < 5</code>	False
<code>></code>	Mayor que	<code>10 > 5</code>	True
<code><=</code>	Menor o igual a	<code>126 <= 100</code>	False
<code>>=</code>	Mayor o igual a	<code>5.0 >= 5.0</code>	True

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

print

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

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

- ***Nota: En Python 2.7 no se usa parentesis.***

print "Message"

print *Expression*

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

input

- input :Lee un numero entrado por el usuario.

- Se puede asignar (almacenar) el resultado de input a una variable.
- Ejemplo(py3)

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

Output:

How old are you? 53

Your age is 53

You have 12 years until retirement

Strings

- **string:** Una secuencia de caracteres textuales en un programa.
 - Strings empiezan y terminan en un character " o apostrophe ` .
 - Ejemplos:
"hello"
"This is a string"
"This, too, is a string. It can be very long!"
- Un string no puede escribirse en varias líneas ni contener un carácter " .
"This is not
a legal String."
"This is not a "legal" String either."
- Un string puede representar caracteres precediéndoles con un backslash.
 - \t tab character
 - \n new line character
 - \" quotation mark character
 - \\ backslash character
- Example: "Hello\tthere\nHow are you?"

Operations on 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")
```

Lists

Es un objeto que puede contener distintos tipos de datos:

A=[0]

B=[2.3, 4.5]

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

Print(type(C))

<type 'list'> # Lista vacia

D=[]

Usar len() para obtener la longitud de una lista

```
>>> names = ["Alicia", "Elena", "Julia"]
```

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

3

Lists (cont)

#Anade un elemento al final de la lista

```
B.append("Alicia")
```

```
B
```

```
[2.3, 4.5, 'Alicia']
```

#Elimina el elemento 4.5 de la lista

```
B.remove(4.5)
```

```
[2.3, 'Alicia']
```

#Elimina el ultimo elemento de la lista

```
C.pop()
```

```
[5, "Hello", "there"]
```

Una tuple es una estructura de datos similar a lista pero cuyos elementos no pueden ser modificados.

Use of [] to index items in a list

```
>>> names[0]  
'Alicia'  
>>> names[1]  
'Camila'  
>>> 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]  
'Camila'  
>>> names[-3]  
'Alicia'
```

[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

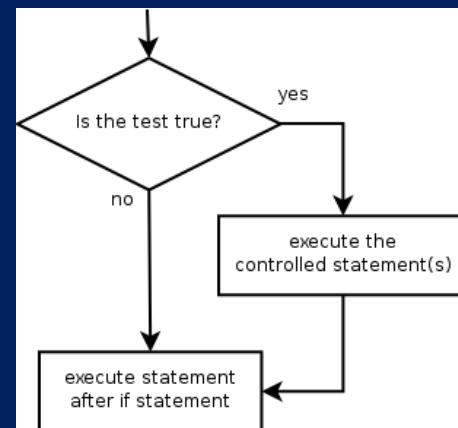
if

- **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("su solicitud es aceptada.")
```



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:

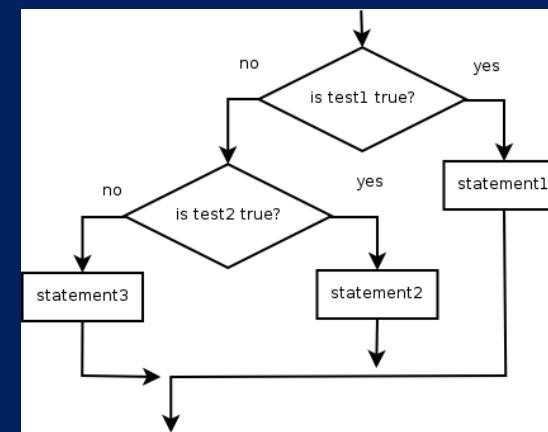
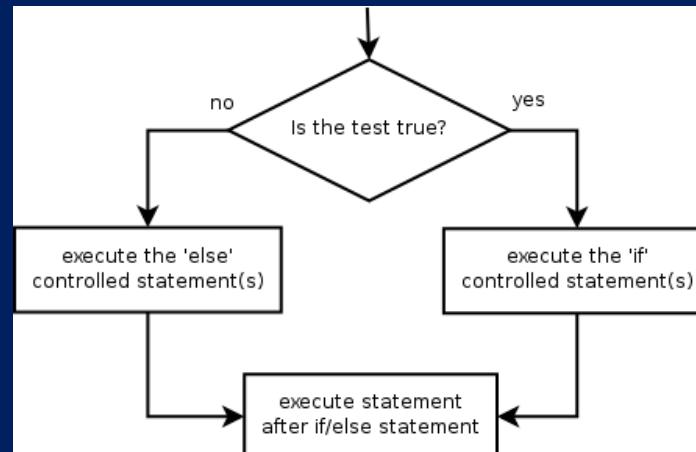
```
if condition:  
    statements  
else:  
    statements
```

- Ejemplo:

```
gpa = 1.4  
if gpa >= 2.5:  
    print("Bienvenido a la la UPR!")  
else:  
    print("Su solicitud es denegada.")
```

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

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



Boolean Logic

Python statements can have “and”s and “or”s:

Alicia=3

Camila=25

if (Alicia <= 5 and Camila >= 10 or

Camila == 500 and Alicia != 5):

 print (“Alicia and Camila”)

Range()

- La funcion “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.

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

Note that the last element generated by range(n) is n-1.
In Python 3, the function “range” works in a different way than en Py2 since Python 3 uses more iterators than lists.

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, para que puedan ser referidos en los enunciados
- **groupOfValues** puede ser un rango de enteros especificados con la función `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
```

for [2]

```
>>> names = ["Alicia", "Camila", "Julia"]
```

```
>>> for name in names:  
...     print (name)
```

```
...
```

Alicia

Camila

Julia

List Comprehension

```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares) # Imprime [0, 1, 4, 9, 16]
```

Otra forma de hacer esto es usando la function map la cual sera vista mas adelante.

Hay otros tipos de containers (objetos para almacenar datos) tales como sets y dictionaries, pero que no seran discutidos aqui.

Break, continue

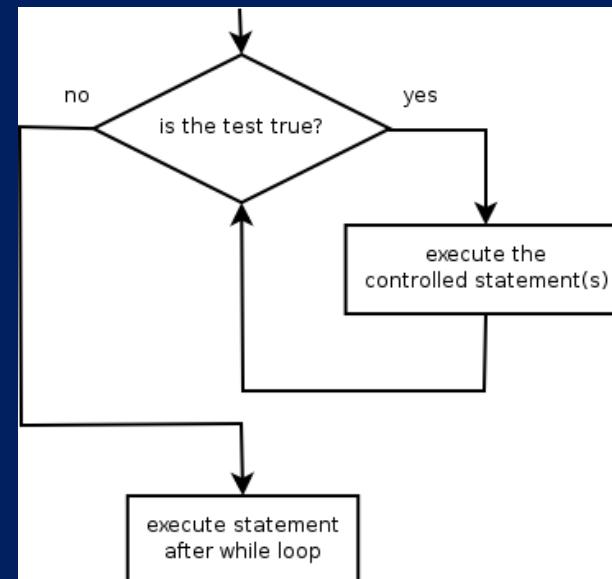
```
>>> for value in [3, 1, 4, 1, 5, 9, 2]:  
...     print ("Checking", value)  
...     if value > 8:  
...         print ("Exiting for loop")  
...         break  
...     elif value < 3:  
...         print ("Ignoring")  
...         continue  
...     print ("The square is", value**2)  
  
Checking 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 condición dada sea cierta.
 - Bueno para loops *indefinidos* (repetir un enunciado un número desconocido de veces)
- Sintaxis:

```
while condicion:  
    enunciado
```
- Ejemplo:

```
number = 1  
while number < 200:  
    print(number),  
    number = number * 2
```



File Input

- Muchos programas manipulan datos, que a menudo estan en files.
- Para leer el contenido de un file en python 3 usar:

```
variableName = open ("filename", 'r')  
variableName = open ("filename") .read()
```

Example:

```
File=open ("c://PW-PR/Animals2.csv") .read()
```

A faster way to read files

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

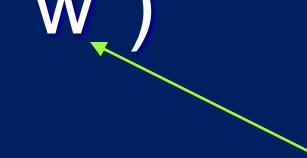
Tambien se puede leer datos directamente de la web usando el modulo urllib.

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”



Modules

- Un programa en Python inicialmente solo tiene acceso a unas funciones basicas.
("int", "dict", "len", "sum", "range", ...)

Para obtener una lista de todas las funciones disponibles.

Ejecutar:

```
for e in __builtins__.__dict__:  
    print(e)
```

- El uso de “Modulos” le anade mas funcionalidad a Python. Para cargar un modulo se usa el comando “import” .

More Modules

- Ejemplos

```
>>> import math #Calculo de funciones matematicas  
>>> import numpy #Contiene calculos estadisticos y de matrices.  
>>>import scipy # Para hacer calculos cientificos 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  
>>> import sklearn #Importa la libreria scikit-learn para hacer Machine learning
```

“import” and “from ... import ...”

```
>>> import numpy as np #usando el alias np para numpy  
np.mean  
>>> from math import cos, pi #importa solo cos y pi de math  
cos #se puede usar directamente cos  
>>> from math import * #importa todas las funciones de math  
Algunos de estos modulos tienen sub-modulos  
>>> import scipy.optimize  
>>> import scipy.stats  
>>> import matplotlib.pyplot as plt
```

Using the module 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)
```

Commands in module math

- Python tiene varios comandos para hacer calculos.

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

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

Functions

Para crear funciones se usa el keyword def

```
def admision(igs):  
    if igs>=300:  
        return 'Admitido'  
    else:  
        return 'Denegado'
```

#Aplicando la funcion a 3 estudiantes

```
for x in [310, 290, 289]:  
    print(admision(x))  
#Una funcion puede tener mas de un argumentos  
def admision1(nombre, igs):
```

```
    if igs>=350:  
        print (nombre, 'fue Admitido' )  
    elif igs<270:  
        print (nombre,'fue Denegado' )  
    else:  
        print(nombre, 'esta en la lista de espera')
```

Map

Funcion map() aplica una funcion dada a cada elemento de un objeto iterable (lista, tuple, etc.) y devuelve una lista de los resultados

La sintaxis es:

map(function,iterable)

```
def cuadrar(n):  
    return n*n  
x = (1, 2, 3, 4)  
res= map(cuadrar, x)  
print(res)
```

Lambdas

Una funcion lambda es una funcion sin nombre que se usa para definir operaciones sencillas.
map(function, iterable)

El ejemplo anterior puede ser ejecutado asi:

```
z = (1, 2, 3, 4)
res = map(lambda x: x*x, z)
print(res)
```

La funcion lambda puede tener mas de un argumento

```
x= [4, 5, 6]
y= [5, 6, 7]
result = map(lambda n1, n2: n1+n2, x, y)
print(list(result))
```

Using Numpy

Numpy es la libreria basica en Python para llevar a cabo tareas de computacion científica . Trabaja con objetos que son considerados arreglos.

Un arreglo en Numpy es un conjunto de valores, todos ellos del mismo tipo, que es indexado por una tupla de vallores nonnegativos. El numero de dimensiones es el rango del arreglo; el shape de un arreglo es un tuple de enteros que da el tamano del arreglo en cada una de las dimesnosne

```
import numpy as np
a = np.array([1, 2, 3])      # Crea un arreglo uni-dimensional
print(type(a))              # Imprime '<class 'numpy.ndarray'>'
print(a.shape)               # Imprime "(3,)"
a[0] = 5                    #Cambia el primer elemento del arreglo a al valor 5
print(a)                     # Imprime "[5, 2, 3]"
b = np.array([[1,2,3],[4,5,6]])  # Crea un arreglo bidimensional
print(b.shape)               # Prints "(2, 3)"
```

Numpy -1

```
a = np.zeros((2,2)) # Crea un arreglo 2x2 de ceros
print(a)
[[ 0.  0.]
 [ 0.  0.]]
b = np.ones((1,2)) # Crea un arreglo 1x2 de unos
print(b)
[[ 1.  1.]]
c = np.full((2,2), 5) # Crea un arreglo 2 x 2 de 5's
print(c)
[[ 5.  5.]
 [ 5.  5.]]
d = np.eye(2)      # Crea una matriz identidad 2x2
print(d)
[[ 1.  0.]
 [ 0.  1.]]
```

Numpy-2

```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
```

```
# Usando slicing para extraer un sub-arreglo consistente de las dos primeras  
# filas y de las columnas 1 y 2  
# b es un arreglo 2x2  
b = a[:2, 1:3]  
print b
```

```
x = np.array([[1,2],[3,4]])  
y = np.array([[5,6],[7,8]])
```

```
print(x + y)  
print(np.add(x, y))  
print(x * y)  
print(np.sqrt(x))  
print(x.T)
```

Pandas

Modulo para Analisis de datos, bastante similar a R.

Estructuras de datos:

Series: Arreglo uni-dimensional

DataFrames: Arreglo Bidimensional

Panel: Arreglo Tridimensional

```
import pandas as pd  
import numpy as np  
data = np.array([8,11,21,13])  
s = pd.Series(data)  
print s  
s.mean()  
s.median()  
s.var()  
s.std()
```

Pandas-1

```
import pandas as pd  
data = [['Renato',19],['Paolo',16],['Camila',13]]  
df = pd.DataFrame(data,columns=['Name','Age'])  
print df
```

Reading data using Pandas

```
# desde un file en su propia computadora
import pandas as pd
train=pd.read_csv('c:/Users/edgar2017/Downloads/titanic.csv')
#directamente de la internet
#na_values representa los valores faltantes(missing values)
#Las columnas no tienen nombres
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=['?'])
```

Modules for graphics

Matplotlib

Seaborn

Bokeh

Plotly

Plotnine

Dash