

Clase 1: Python Basics para Data Analytics

Manifesto

Estas clases tienen por objetivo ser un apoyo/referencia para comenzar a trabajar con Python en problemas de Data Analytics. Para esto, el plan de estudio contempla:

- ✓ Comprender el Universo de Python
- ✓ Fundamentos de Python y sus Estructuras de Datos
- ✓ Numpy
- ✓ Uso de *Scikit-learn* en aprendizaje supervizado

“Como todo lenguaje de programación las posibilidades son ilimitadas, por ello el límite de aprendizaje depende de cada uno, lo que yo puedo enseñar es limitado y lo que ustedes pueden aprender es ilimitado. El instructor no es un mago que puede transferir conocimientos por osmosis forzada ni nada por el estilo. También recuerda que el instructor no tiene la respuesta a todas tus preguntas.”

Autor: Jorge Felipe Monardes Pinto

jorge.monardes.pinto@everis.com
jorge.monardes@duke.edu

Clase 1: Python Basics para Data Analytics



an NTT DATA Company

Fuente

Visitar mi repositorio Github : <https://github.com/juxtap>

Documentación y scripts : https://github.com/juxtap/py3_capacitacion

The screenshot shows Jorge Monardes' GitHub profile. At the top, there's a search bar, a navigation bar with 'Pull requests', 'Issues', and 'Gist', and a user menu. Below the header is a large profile picture of Jorge, a young man with dark hair, smiling and giving a thumbs-up. To his right are three tabs: 'Overview' (selected), 'Repositories', and 'Public activity'. On the far right is an 'Edit profile' button. Under 'Popular repositories', there are two pinned repositories: 'cs571D' (Machine Learning with Python) and 'py3_capacitacion' (Capacitación en español de Python básico). Below this is a section titled '5 contributions in the last year' with a timeline from September to August.

Jorge Monardes
juxtap

Popular repositories

- cs571D**
Project consisting on Machine Learning with Python, a brief review about scikit-learn.
- py3_capacitacion**
Capacitación en español de Python básico con simples y entretenidos scripts de ejemplo

5 contributions in the last year

Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug



jorge.monardes.pinto@everis.com
jorge.monardes@duke.edu



Context: Language is not the Goal

- Language is a means not the goal

- Learning new computer languages is easy
- Much easier than natural languages
- So, come with confidence!

Más simple !



<http://www.dreamstime.com/royalty-free-stock-photography-3d-knob-confidence-level-image29083987>

CompSci 201

```
int sum = 0;  
for(int i=0; i<100; i=i+1) {  
    sum = sum + i;  
}
```

* Los lenguajes de programación
fueron diseñados en inglés

lebih mudah daripada bahasa semula jadi
قدرتى زبانوں کے مقابلے میں آسان
קל יותר משפות טבעיות
比自然语言更容易
ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਵਿਖੇ ਅਸਾਨ ਹੈ

Episode 01.19

Clase 1: Instalar Python

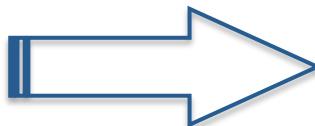
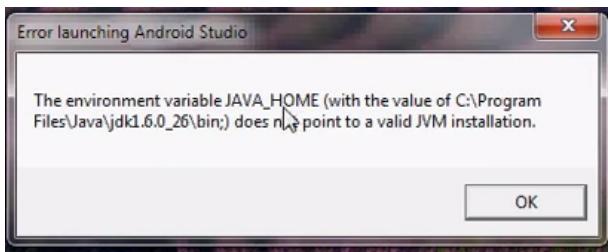
Para instalar Python y recursos recomendados:

- Visitar <https://www.python.org> Download -> Python 3.5.X
- Instalar SciPy Stack: <https://www.continuum.io/downloads>.

Para usuarios Windows/Linux seleccionar Python 3.5 versión, 64-BIT INSTALLER. Para usuarios OSX seleccionar Python 3.5 versión -> GRAPHICAL INSTALLER.

- Instalar PyCharm: <https://www.jetbrains.com/pycharm/download/>

Los usuarios de Windows puede que experimenten problemas con la ruta de las variables de entorno JAVA_HOME:



<https://www.youtube.com/watch?v=GwU4AJn0Txg>
(ver video en caso de problemas)

¿ Python 2.x o Python 3.x ?

“Python 2.x is legacy, Python 3.x is the present and future of the language.”



Clase 1: Universo

Creado por : Guido van Rossum en 1990

NLP, ¿low-level?

Variedad de aplicaciones



No-compiling

Usos : "High-level, general-purpose, interpreted, dynamic programming language."

Runtime executes many tasks that static languages perform during compilation.

Paradigmas : "Object-oriented, imperative, functional/procedural programming and reflective."

Ability of a computer program to examine, introspect, and modify its own structure and behavior during runtime

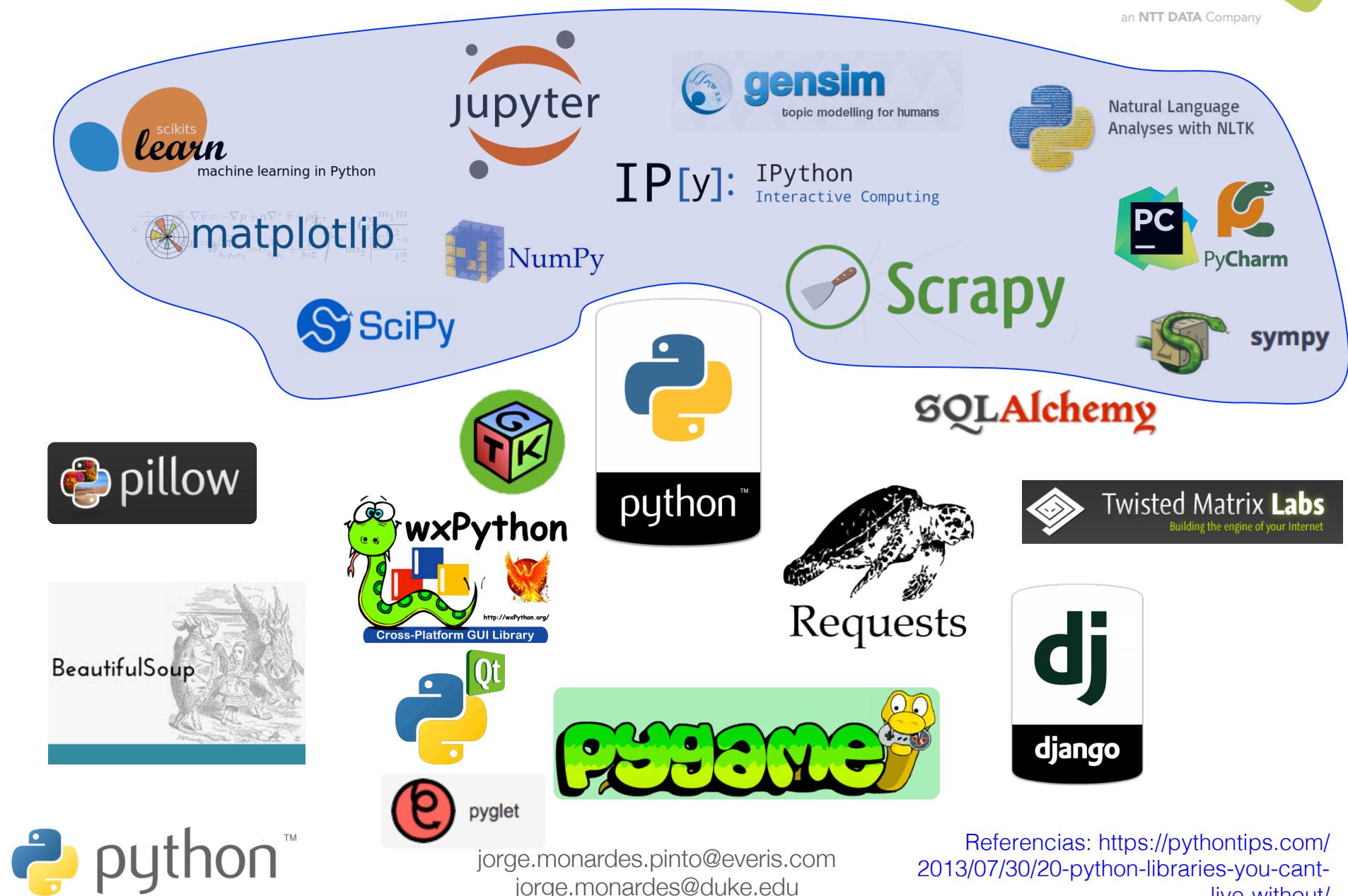
methods, ..., +methods

Focused on *How* (statements) vs. declarative programming focused on *What*. (e.g. SQL)

*ontologies, cognitive science

Clase 1: Universo

an NTT DATA Company



Clase 1: Fundamentos – Primitive Types



Java

- ✓ int myInt = 10;
- ✓ char myChar = 't';
- ✓ boolean myBool = false; //o true
- ✓ double myDouble = 10.5;

TNV

Python

- ✓ myInt = 10
- ✓ myChar = 't' //o "t"
- ✓ myBool = False //o True
- ✓ myDouble = 10.5
- ✓ myFloat = 10.5

Nombre Valor

Clase 1: Fundamentos – Operadores

Operación	Resultado
$x + y$	Suma de x más y
$x - y$	Resta de x menos y
$x * y$	Producto x por y
x / y	Cociente x dividido en y
$x // y$	Piso del cociente
$x \% y$	Remanente de la división x / y
$\text{abs}(x)$	Valor absoluto de x
$\text{int}(x)$	Conversión de x a Z
$\text{long}(x)$	Conversión ...
$\text{float}(x)$	Conversión de x a R

Operación	Resultado
$\text{complex}(re, im)$	Número complejo, re parte Real y im parte imaginaria
$c.\text{conjugate}$	Conjugado del complejo "c" (ej: $3 + 2i \Rightarrow 3 - 2i$)
$\text{pow}(x, y)$ o $x^{**} y$	x exponente y

Revisar más en librería math:

<https://docs.python.org/2/library/math.html>

Clase 1: Fundamentos – Booleanos y Operadores

Operación	Resultado
<code>x = True</code>	x es verdadero
<code>x = False</code>	X es falso
True and False	False
True & False	False
True or False	True
True False	True
<code>x < y</code>	x menor estricto que y
<code>x <= y</code>	x menor igual que y
<code>x > y</code>	x mayor estricto que y
<code>x >= y</code>	x mayor igual que y

Operación	Resultado
<code>x == y</code>	x igual a y
<code>x != y</code>	x distinto de y
<code>x is not None</code>	True (si x = 10)

```
>>> x = 10 if True & False else 3
>>> x
3
>>> x = 10 if True | False else 3
>>> x
10
>>> x = 10 if True and False else 3
>>> x
3
>>> x = 10 if True or False else 3
>>> x
10
>>>
```

Clase 1: Fundamentos – Strings

Sintaxis : “ ____ ” o ‘ ____ ’

myString = “Hello world”

miString = ‘Hola mundo’

```
print(myString + “*español: ” + miString)
```

```
>>> Hello world*español: Hola mundo
```

Método más utilizado -> len()

```
len(miString)
```

```
>>> 10
```

Revisar métodos para Strings en:

<https://docs.python.org/2/library/stdtypes.html#string-methods>

```
print(miString + “, número de caracteres”  
+ len(miString))
```

>>> Error

```
print(miString + “, número de caracteres”  
+ str(len(miString)))
```

>>> Hola mundo, número de caracteres 10

Clase 1: Fundamentos – Listas y Diccionarios

Listas:

myList = list() o myList = []

myList.append("hola")

>>> ["hola"]

myList.append(8)

>>> ["hola", 8]

myList.append(True)

>>> ["hola", 8, True]

Considerar Listas como vectores de objetos.

Diccionarios:

myDiccionary = { key1: value1,
key2: value2, ..., keyN: valueN }

También, myDiccionary = {}

*Considerar como un
HashMap o
HashTable*

myAges = {"Jorge": 30, "Paula": 25,
"José": 28, "Andrea": 42, "Nuria": 26}

myAges["Paula"]

>>> 25

myAges["Martin"] = 38

myAges

>>> {"Jorge": 30, "Paula": 25, "José":
28, "Andrea": 42, "Nuria": 26, "Martin":
38}

Clase 1: Fundamentos – Condicionales

if expresión booleana:

statement T

else:

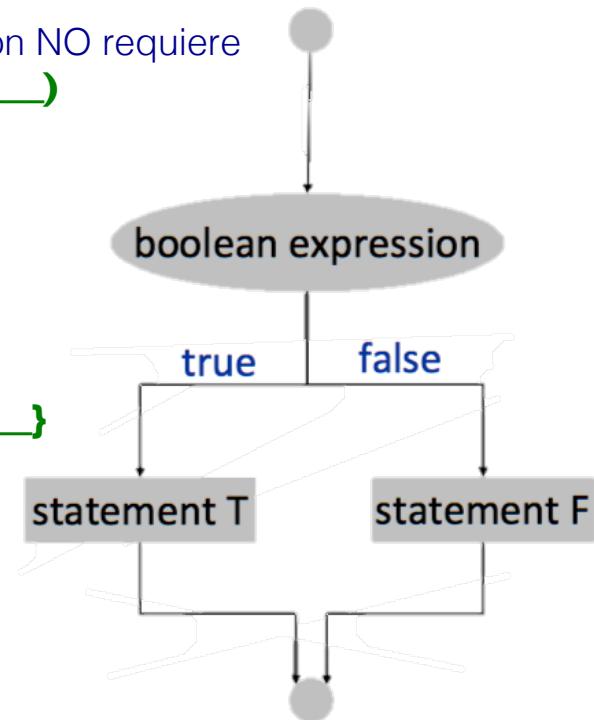
statement F

OJO_2 : El TAB en Python es importante y no requieres {} para bloques de códigos TAB equivale a cuatro espacios

Elseif, también es una alternativa para los flujos de control, en Python es **Elif**, Nesting también es posible.

El equivalente a un **switch** no existe en Python, pero es fácil de generar esta estructura utilizando un diccionario de funciones o una función diccionario.

OJO_1 : Python NO requiere paréntesis if()



C# , PHP, VB

Clase 1: Fundamentos – Condicionales

Estructura para **Elif** :

if expresión booleana:

 statement 1

elif expresión booleana:

 statement 2

elif expresión booleana:

 statement 3

...

else:

 statement N

Nesting:

if expresión booleana:

if expresión booleana:

 statement 1

else:

if expresión booleana:

 statement 2

else:

if expresión booleana:

 statement 3

else:

...



Clase 1: Fundamentos – Loops (**for** __)

an NTT DATA Company



for loop sintaxis:

for num in myNumbers:

statement 1

...

statement N

```
myNumbers = range(10)  
          = [0, 1, ..., 9]
```

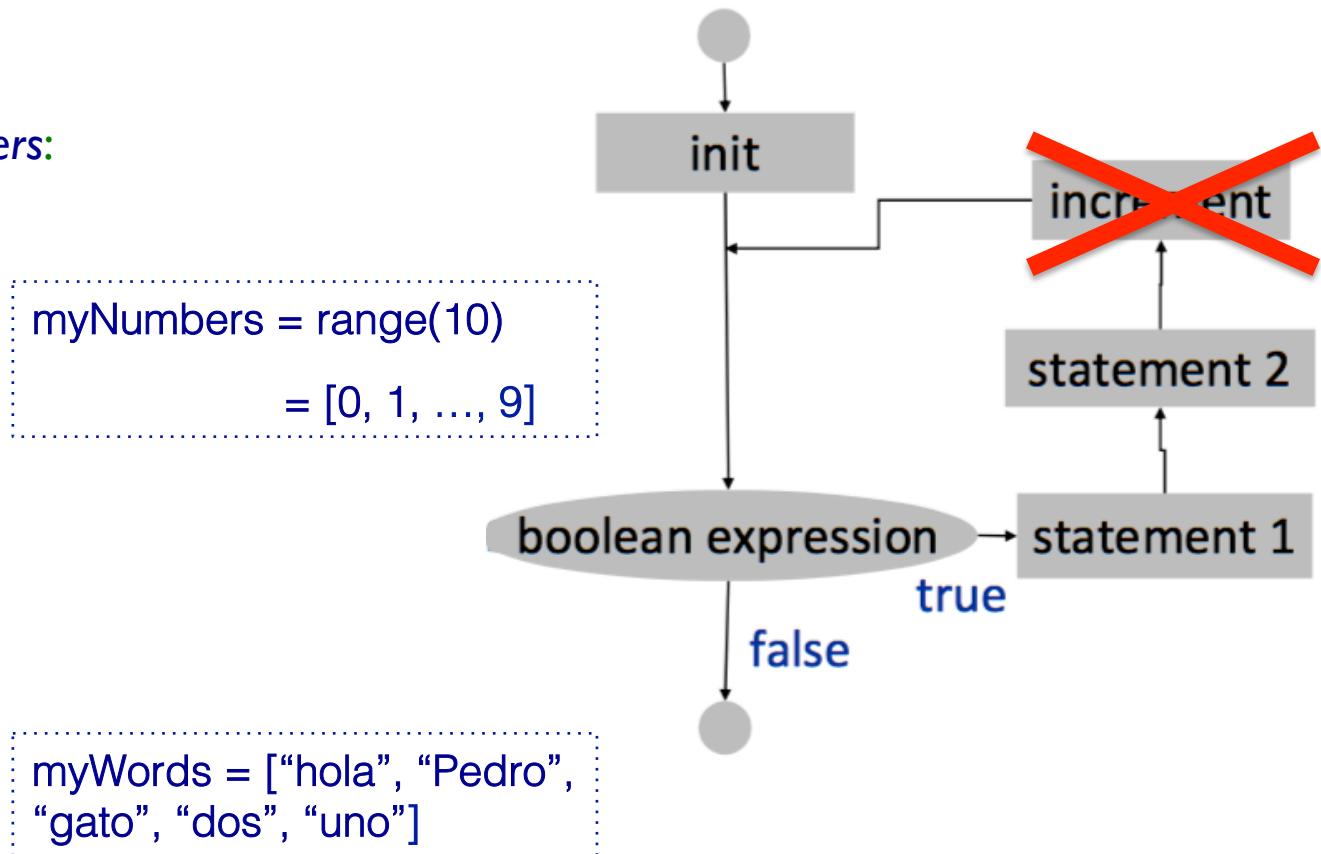
for w in myWords:

statement 1

...

statement N

```
myWords = ["hola", "Pedro",  
          "gato", "dos", "uno"]
```



Los loops de Python soportan **break**, **continue** y **pass**.



jorge.monardes.pinto@everis.com
jorge.monardes@duke.edu

Referencias: Prof. Salman Azhar

Clase 1: Fundamentos – Loops (**while** __)

an NTT DATA Company



while loop sintaxis:

while expresión booleana:

statement 1

...

statement N

Ejemplo simplificado

x = 0

myList = list()

while x < 5:

 myList.append(x)

 x += 1

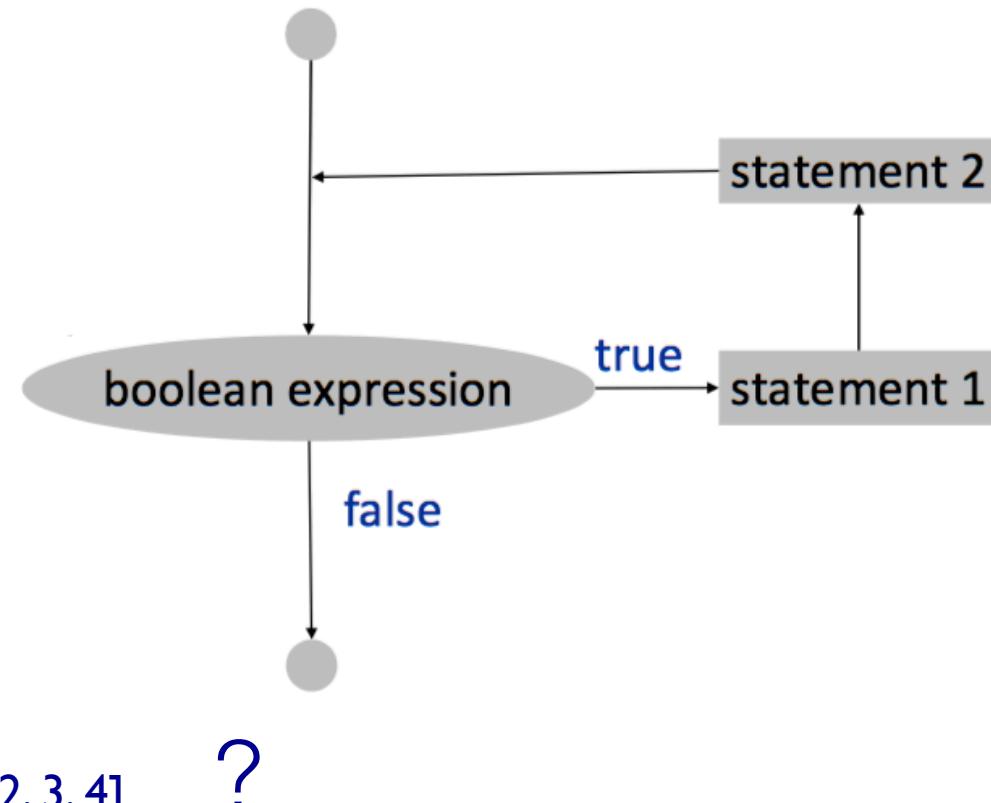


myList = [0, 1, 2, 3, 4]

?

=

myList = [x **for** x **in** range(5)]



jorge.monardes.pinto@everis.com
jorge.monardes@duke.edu

Referencias: Prof. Salman Azhar

Clase 1: Fundamentos – Funciones

an NTT DATA Company

“Las funciones (métodos) son simplemente bloques de código que pueden ser reutilizados para cumplir con una tarea/rutina de programación.”

Sintaxis de funciones en Python **def** :

def functionName(arguments):

statement 1

...

statement N

return _____

(opcional)

Simple! No hay necesidad de definir si es public/private, type: int, long, String, int[], String[], et cetera.

Ejemplo

import math

def cylinderVolume(r ,h):

baseSurface = (r ** 2) * math.pi

return baseSurface * h

Statement **import** es para importar librerías. *math* es una de las librerías básicas mas utilizadas en Python.



jorge.monardes.pinto@everis.com
jorge.monardes@duke.edu

Referencias: Code Academy – Python

Clase 1: Fundamentos – Clases

Las clases son estructuras de datos que nos permiten definir la estructura (variables y métodos) correspondientes a objetos que pertenecen a dicha clase.

Sintaxis de Clases en Python **class**:

class *ClassName*:

statement(s)

def *__init__*(*self*, *args*):

self.arg1 = *arg1*

self.agr2 = *arg2*

 ...

self.argN = *argN*

def *method1*(*self*):

statement(s)

def *method2*(*self*, *args*):

statement(s)

No hay public/ private o local/ global explícitos, pero sí existen implícitamente !

Constructor

Las convenciones actualmente utilizadas para variables/métodos non-public son:

_variable = ...

def *__function*(*args*):

Revisar mangling y private variables.

Ejemplo

class *EverisCollaborator*:

company = ‘Everis NTT Data’

def *__init__*(*self*, *name*, *skills*, *sleepingHours*):

self.name = *name*

self.skills = *skills*

self.sleepingHours = *sleepingHours*

def *profile*(*self*):

print “nombre: %s, principal habilidad: %s, horas diarias de sueño %d ” % (*self.name*, *self.skills*[0], *self.sleepingHours*)

def *addSkill*(*self*, *skill*):

self.skills.append(skill)

Referencias: <https://docs.python.org/3/tutorial/classes.html>

Clase 1: Fundamentos – Clases

class EverisCollaborator:

```
company = 'Everis NTT Data'  
def __init__(self, name, skills, sleepingHours):  
    self.name = name  
    self.skills = skills  
    self.sleepingHours = sleepingHours  
def profile(self):  
    print "nombre: %s, principal habilidad: %s,  
horas diarias de sueño %d horas" % (self.name,  
self.skills[0], self.sleepingHours)  
def addSkill(self, skill):  
    self.skills.append(skill)
```

Si estas interesado en estilo y buenas prácticas de programación en Python, revisa estándares y buenas prácticas en
<https://www.python.org/dev/peps/pep-0008/>

Creemos objetos con nuestra clase:

```
felipe = EverisCollaborator("Felipe Monardes",  
["python", "java", "php"], 5)
```

```
felipe.company
```

```
>>> Everis NTT Data
```

```
print(*felipe.skills)
```

```
>>> python, java, php
```

```
felipe.addSkill('finance')
```

```
felipe.addSkill("teaching")
```

```
felipe.profile()
```

```
>>> nombre: Felipe Monardes, principal habilidad  
python, horas diarias de sueño 5 horas
```



```
felipe.name = "Jorge Monardes"
```

```
felipe.sleepingHours = 7
```

Referencias: <https://docs.python.org/3/tutorial/classes.html>

Clase 1: Fundamentos – Text I/O

Python nos permite de manera fácil y rápida escribir/leer archivos de extensiones *.txt, *.csv, *.xlsx, et cetera.

Sintaxis para escribir archivo

```
my_odd_nums = [x for x in range(101)  
               if x % 2 != 0]  
  
file = open("numeros_impares.txt", "w")  
  
for num in my_odd_nums:  
    file.write(str(num) + "\n")
```

```
file.close()
```

Cerrar doc !!!!

SIEMPRE

Generación de listas en base a condicionales

Abro puntero para solo escritura "w"

Escribo elemento por elemento y salto de línea ASCII

"r" : Solo lectura

"r+": lectura y escritura

Clase 1: Fundamentos – Text I/O



Sintaxis para escribir leer CSV

```
import csv
```

```
file = open('inputFile.csv', 'r')
```

```
pointer_csv = csv.reader(file)
```

```
doc = list()
```

```
temperature = []
```

```
row_num = 0
```

```
for row in pointer_csv:
```

```
    if row_num == 0:
```

```
        header = row
```

```
    else:
```

```
        doc.append(row)
```

```
        temperature.append(row[4])
```

```
    row_num += 1
```

```
file.close()
```

Crear objeto (puntero de lectura) utilizando **librería csv**

Variable **doc** será encargada de guardar el documento completo de **forma matricial**

Supongamos que la columna 5 de “inputFile.csv” corresponde a un **vector de temperatura** y ese es el valor más importante que deseamos guardar.

Comenzamos leyendo desde la **primera** fila hasta la **última**

Guardamos la primera fila como las **etiquetas** de cada columna

Guardamos documento completo + variable de temperatura

CERRAMOS archivo !

Clase 1: Numpy, Scikit-learn, Matplotlib

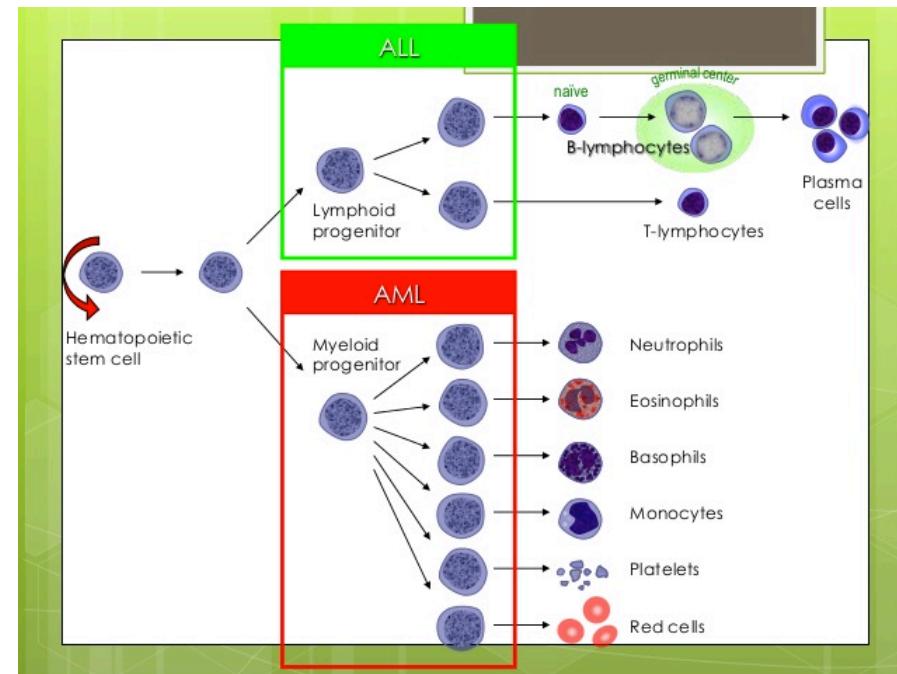
Ejemplos durante la clase !!



Clase 1: Conceptualización ejemplo 1 -- Leucemia

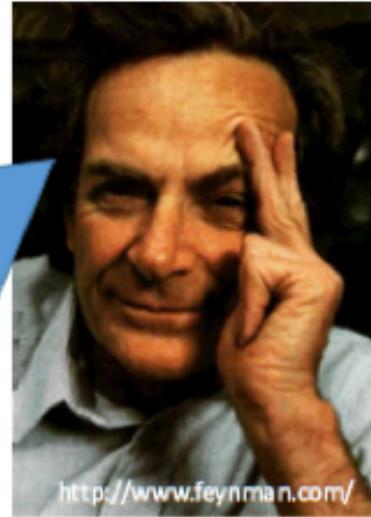
Anualmente se desarrollan 352,000 casos de leucemia en el mundo, causando 265,000 muertes. Es el tipo de cáncer más común en los niños y niñas, correspondiendo a más del 75% de los casos.

ALL (Acute Lymphocytic Leukemia) y AML (Acute Myelogenous Leukemia) son los dos tipos más frecuentes en menores. Identificar si el menor posee ALL o AML resulta sumamente importante para diagnosticar tratamiento y cantidad de dosis en el paciente.



Freedom

So I have just one wish for you - the good luck to be somewhere where you are **free to maintain** the kind of **integrity** I have described, and where you do not feel forced by a need to maintain your **position** in the organization, or **financial** support, or so on, to lose your integrity. May you have that **freedom**.



<http://www.feynman.com/>

Richard Feynman
(1995 Noble Prize
in Physics)