Introducción Lenguage Python

Temario de la clase

- Características de python.
- Porque programar en python? Cuando si y cuando no?
- Porque los pythónicos son tan entusiastas?
- Python como una calculadora.
- Librerías.
- Variables. Tipos de variables

Que es Python?

Lenguaje de muy alto nivel (sintaxis comprensible y muy sencilla).
Lenguaje interprete (No necesita de compilación).
Lenguaje estructurado. La tabulación es parte de la sintaxis.
Lenguaje orientado a objetos.
Lenguaje utilizado en la mayoría de las aplicaciones desarrolladas por Google.
Youtube esta hecho en python.
Lenguaje utilizado por Industrial Light & Magic en la producción de Star Wars y por DreamWorks Animations.

Porque Python?

Programas muy compactos (3-4 veces mas corto que en fortran o C).
Programas legibles.
Lenguaje estructurado y orientado a objetos.
Es un lenguaje open-source (gnu).
Muy fácil /de integrar con/integrador de/ otros lenguajes C/Fortran/Java.
Enorme comunidad de desarrolladores y usuarios, también en el ámbito científico.
Programar disfrutando!

Cuando no usar Python?

Python es un lenguaje de muy alto nivel, eso trae aparejado ventajas y desventajas.
Entre las segundas encontramos:
La ejecución de fórmulas matemáticas intensivas (bucles de > 10000 ciclos) suele ser lenta. Para computación numérica de alta performance el mas eficiente es Fortran seguido de C.
El control de dispositivos de la computadora (hardware) es indirecto e insuficiente. El C es el lenguaje con el mayor control de dispositivos en forma eficiente (Los SOs estan en C!).
La estructura flexible puede ser peligrosa si no se siguen reglas. (Un mal uso puede llegar a bugs difíciles de rastrear).

Python en la física

Existe una enorme cantidad de librerías científicas en Python todas ellas disponibles como open-source.
Desde procesos numéricos básicos: ej. integración, matrices.
Librerías de graficación en 3D.
Librerías para el tratamiento estadístico de datos.
Librerías de inteligencia artificial.
Muchos de los programas que desarrollamos en el grupo de investigación son adaptaciones o utilizan librerías ya desarrolladas por otros grupos.

Para que usamos nosotros el python?

1. Todo tipo de prueba o test sencillo que se quiera realizar en python.
2. Desarrollo de métodos y algoritmos: se comienza en python y luego se lleva a fortran
3. Las simulaciones se realizan en fortran (cluster) pero luego el análisis lo realizamos en python, la graficación en python (o idl).
4. Desarrollo de software para empresa Claro. Manejo estadístico. Todo python.

La manera pythonica de programar

Reglas para programar en python. Comando

```
>>> import this
```

- Lindo es mejor que feo
- Explícito es mejor que implícito
- Simple es mejor que complejo.
- Directo es mejor que anidado.
- Esparso es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son tan especiales como para romper las reglas
- El pragmatismo por sobre la pureza
- Los errores deben siempre hacerse notar.
- Debería haber una y preferiblemente solo una forma obvia de hacerlo.
- Ahora es mejor que nunca.
- Si la implementación es difícil de explicar, entonces es una mala idea.

Como usar python

En una terminal shell/bash. Desde línea de comando:

```
$ python
```

Si se quiere ejecutar en forma remota (en otra computadora)

```
$ ssh usuario@computadora
```

```
$ ssh usuario@200.45.54.94
```

Luego se puede ejecutar el python:

```
$ python
```

Manejo de ventanas múltiples remotas

En una terminal shell/bash o desde el putty.

```
$ ssh usuario@200.45.54.94
```

Desde línea de comando:

```
$ screen
```

- Si quieren crear una nueva terminal: Ctrl + a + c
  - Pueden tener una terminal con un nano (archivo que estan editando).
  - Una ventana en bash. Y una tercera ventana en python.
- Si se quiere volver a la terminal anterior: Ctrl + a + p
- Si se quiere ir a la terminal siguiente: Ctrl + a + c

Entonces hagan screen. Habran una nueva terminal Ctrl + a + c y ejecuten python.

Python como una calculadora

Operaciones aritméticas

```
>>> 55+15
>>> -33+12
>>> 5.2/3.1 + 2
>>> 4**2
>>> 4**0.5
```

Python como una calculadora. Tipo de variables.

```
>>> 5/2
>>> 5/2.0
>>> 3.0e3
>>> 'Hola'
```

Tipos de varibles: Enteros. Flotantes. Cadena de caracteres. Lógicas. Números complejos

```
>>> a=5+2j
>>> type(a)
>>> print a.real
>>> print a.imag
```

Transforma un número flotante/entero en complejo:

```
>>> complex(5.0)
```

Python como una calculadora. Orden de las operaciones.

Orden de las operaciones aritméticas.

```
>>> 5*3**2
>>> (5*3)**2
>>> 5+3**2
>>> (5+3)**2
```

1. $**$, 2. $*, /$, 3. $+, -$. Se altera con $()$

Variables

Supongamos que queremos tener disponible el valor de π para usarlo en varias ocasiones, en lugar de tener que tipearlo cada vez que lo necesitemos, entonces asignamos a una variable el valor de pi:

```
>>> pi=3.14159
```

Luego la variable pi tiene guardado el valor 3.14159, compruebelo poniendo:

```
>>> pi
```

Si ahora queremos calcular el perímetro de una circunferencia de radio 20cm,

```
>>> 2*pi*20
```

Quizas también nos convenga guardar el radio en una variable:

```
>>> radio=20
```

la circunferencia es entonces:

```
>>> 2*pi*radio
```

y la superficie de la circunferencia es:

```
>>> radio*pi**2
```

Variables. Reasignación

Si durante la ejecución necesitamos reasignar el valor de la variable, supongamos ahora tenemos una circunferencia de 15.5cm, redefinimos la variable:

```
>>> radio =15.5
>>> radio
>>> 2*pi*radio
>>> radio*pi**2
```

La variable ha tomado el nuevo valor, notar además que cambiamos el tipo de variable, al principio la definimos como entera y luego como flotante.

Salidas por pantalla: print

Para que la computadora ponga como salida algo que nosotros queremos mostrar se usa el comando print.

```
>>> print ('Hola')
>>> radio=2.02
>>> unidad_radio='cm'
>>> print ('El radio es de ',radio,unidad_radio)
```

Si queremos imprimir varias variables y/o cadenas de caracteres las separamos por comas.

En python2 se podía usar

```
>>> print 'Hola'
```

.

En python3 se deben agregar los paréntesis.

A las instrucciones que le damos a la computadora le llamamos comandos o instrucción, `print` es un comando.

Primer programa python

La mayoría de las veces lo que tendremos son muchas líneas de código por lo que queremos guardarla para usarla a futuro. Para esto al código lo tenemos que guardar en un archivo que denominamos “programa”.
Los nombres de los archivos python tienen extensión .py.
Se debe usar un editor: emacs o nano.
Editemos un archivo:

```
$ nano simple.py
```

```
#!/usr/bin/env python
pi=3.141592
radio=20.0
per=2*pi*radio
sup=pi*radio**2
print (’El perimetro es: ’,per)
print (’La superficie es: ’,sup)
```

Ejecución del primer programa python

Para ejecutar un programa python lo que hacemos en una terminal shell/bash es:

```
$ python simple.py
```

Si el programa se encuentra en otro directorio se le da el camino completo:

```
$ python /home/pulido/curso/programacion/pyt/simple.py
```

Si el programa lo han hecho ejecutable (con el

```
chmod +x simple.py
```

) entonces:

```
$ ./simple.py
```

Comentarios en el programa

A menudo queremos comentar las líneas de programa para luego recordar lo que hicimos.

Para explicar lo que hacemos en el código a otro programador.

Para referenciar el objetivo, que es lo que hace, cuando lo hicimos, cuando lo modificamos, que cosas necesitamos agregarle, etc.

Los comentarios de una sola línea se hacen con #:

```
# (todo lo que sigue detrás del símbolo python lo interpretará como un comentario)
```

```
5+8 # suma
```

Comentarios en el programa

Comentarios de varias líneas se hacen con: """ (triple comillas)

```
""" Programa para el calculo del
perimetro y la superficie de una circunferencia.
MP. [2016-08-10]
TODO. Agregar el volumen de una esfera
"""

pi=3.141592
radio=20.0
# Calculo del perimetro
per=2*pi*radio
sup=pi*radio**2 # superficie
print ('El perimetro es: ',perim)
print ('La superficie es: ',sup)
```

REGLA DE ORO 1: Es esencial que todo programa este comentado hasta el último detalle.

REGLA DE ORO 2: Las variables deben tener nombres representativos de la información que contiene e.g. sup (guarda la superficie). Mejor

Entrada de valores

Supongamos que queremos diseñar un programa para estudiantes de la primaria (que no saben programar) que midan el diámetro de distintos objetos circulares y el programa que les calcule el perímetro y la superficie.
El programa les debería pedir que introduzcan el valor del diámetro.
El comando que detendrá la ejecución y pedirá para que el usuario ingrese una cantidad es el input, en general el valor que se ingrese se guardará en una variable:

```
d=input("introduzca el diametro del objeto: ")
```

También podemos ingresar valores

Entrada de valores

Entonces el programita para los estudiantes de primaria con el input sería:

```
pi=3.141592
d=float(input("introduzca el diametro del objeto: "))
# Calculo del perimetro
per=pi*d
sup=pi*(d/2.)**2 # superficie
print ('El perimetro del objeto es: ',per)
print ('La superficie del objeto es: ',sup)
```

A este programa le faltan chequeos para cuando el usuario introduce algo erroneamente.

Print con formato de salida

Supongamos que tenemos un examen y queremos informar los estudiantes presentes

```
print ('La cantidad de asistentes al examen fue de %d alumnos de un curso de %d alumnos. Los ausentes fueron %d' %(nasis,ncurso,ncurso-nasis))
```

Las notas se deben informar con uno o dos dígitos pero el promedio con dos decimales.

```
print ('Perez, Juan        %d' %(nota1))
print ('Sanchez, Pedro     %d' %(nota2))
print ('Promedio           %6.2f' %(0.5*(nota1+nota2)) )
```

%d se utiliza para enteros.

%f para flotantes. %(digitos).(decimales)f

%s para una cadena (string).

Información: help

Esto nos da información de toda la librería, si queremos consultar información de un comando:

```
>>> help(input)
```

```
>>> help(print)
```

Para salir del interprete o terminar un programa:

```
>>> quit()
```

Este es mas simple en el interprete o en un programa.

```
raise SystemExit
```

Este es mejor en un programa.

Operaciones con cadenas de caracteres

Subcadenas [i:j]:

```
>>> sa='cadena'
>>> print ( sa[2:4] )
de
```

Transforma un número en cadena, str

```
>>> a=1239
>>> b=str(a)
>>> print ( b[2:4] )
39
```

Concatena una cadena: +

```
>>> nombre='Juan'
>>> apellido='Perez'
>>> nombre_completo=nombre+' '+apellido'
>>> print ( nombre_completo )
Juan Perez
```

Operaciones con cadenas de caracteres

Cambia a mayúsculas: .upper()

```
>>> a=’casa’
>>> print ( a.upper() )
CASA
```

Reemplaza un caracter o cadena de caracteres: .replace(viejo,nuevo)

```
>>> print ( a.replace(’a’,’o’) )
coso
```

Busca un caracter o cadena de caracteres: .find(’o’)

```
>>> a=’casona’
>>> print ( a.find(’o’) )
3
```

Tipos de variables: listas

Ya hemos visto los tipos: Enteros. Flotantes. Cadena de caracteres. Lógicas.

Pero también existen tipos de variables que son conjuntos de enteros, flotantes, etc...

Una lista es un conjunto de elementos de cualquier tipo separados por comas y delimitado por corchetes:

```
>>> lista=[25,60.4,'edad y peso']
```

Para acceder a un elemento de la lista:

```
>>> print ( lista[1] )
```

Para acceder a varios elementos de la lista:

```
>>> print ( lista[1:2] )
```

Cantidad de elementos de la lista:

```
>>> print ( 'Longitud: ',len(lista))
```

Si quiero cambiar la edad en la lista:

```
>>> lista[0]=26
```

Operaciones con listas

range(<inicio,>fin<,salto>): crea una lista de enteros, de uno en uno (el fin esta excluido!).

```
>>> print ( range(4) )
[0, 1, 2, 3]
```

```
>>> print ( range(2,10,2) )
[2, 4, 6, 8]
```

lista.append(elemento) Agrega elementos a una lista

```
>>> z = [1,2.02]
>>> z.append(800.8)
>>> z
[1, 2.02, 800.8]
```

Operaciones con listas

Las listas tienen funciones muy útiles:

`len(lista), max(lista), min(lista),` siendo `lista` una lista.

Mapa:

`map(funcion,lista)`: aplica la función `funcion` a todos los elementos de la lista `lista`

```
»> lista = ['Ricardo', 'RODOLFO', 'sergio' ]
»> list(map (str.lower,lista))
['ricardo', 'rodolfo', 'sergio']
```

Operaciones con listas

lista.reverse(): invierte la lista

```
>>> z.reverse()
>>> print ( z )
[800.8, 2.02, 1]
```

lista.remove(x): elimina el primer elemento que coincide con x de la lista

lista.pop(j): elimina el elemento j-esimo de la lista

Tipos de variables: tuplas

Las tuplas son secuencias de objetos como las listas pero no se pueden cambiar

```
>>> tupla=(25,60.4,'edad y peso')
```

si engordo y quiero cambiar el peso:

```
>>> tupla[1]=61.6
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

La razón de su existencia es que son mucho mas eficientes sin embargo nosotros la usaremos muy poco (en general requerimos de estructuras mas dinámicas).

Tipos de variables/objetos: Diccionarios

Almacenan un conjunto de variables u objetos.

- Se accede a sus elementos a partir de claves principales (keys).
- La clave sirve para identificar al elemento.
- Se definen con llaves (en lugar de corchetes que es para las listas).

```
>>> x={'nom','dom'}
```

Asignación:

```
>>> x={'nom':'Sergio','dom':'Libertad 5400'}
```

```
>>> x['nom']='Sergio'; x['dom']='Libertad 5400'
```

Funciones para diccionarios: `len, x.keys(), x.values()`

Uso de librerías

Si queremos calcular el logaritmo de un número, será el comando “log” o el seno “sin”?. Si probamos con log(5.) python no entiende y dice que es un error de sintaxis.

El lenguaje python (el nucleo) es extremadamente pequeño, todas las funcionalidades vienen a través de librerias, y la mayoría de éstas estan escritan en python.

Para cargar una librería debemos utilizar el comando import

```
>>> import math
```

Todas los comandos de la librería se pueden acceder por:

```
>>> math.log(5.0)
```

Formas de cargar una librería

Si queremos renombrar (por ejemplo para que el nombre sea mas corto):

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

```
>>> m.log(5.0)
```

Si queremos importar solo un código que esta dentro de una librería:

```
>>> from math import log
```

```
>>> log(5.0)
```

Notar que con esto reconoce el comando directamente. Pero es mas caro y no es aconsejable en programación pero si para hacer pruebas. La forma pythónica es usando objetos.

Uso de librerías

Para evaluar funciones matemáticas de números complejos esta la librería cmath

Si queremos hacer un gráfico, cargamos la librería de graficación pylab.

```
>>> from pylab import *
>>> plot (range (10))
>>> show()
```

Librería numérica de derivadas, integrales, raíces etc: numpy

Librería de graficación completa matplotlib

Librería científica, estadística: scipy BLAS, LAPACK, fft, cluster, etc.

Librería de inteligencia artificial (para big data, data science): scikit-learn

Librería math. Funciones matemáticas

La librería math posee todas las funciones matemáticas estandards: log, trigonométricas: sin, cos, tan, función error, factorial, etc.

Además posee constantes: e, π.

De pasaje de radianes a grado:

```
>>> math.degrees(2.*math.pi)
```

Entonces con el python solito tenemos una calculadora de las truchas, con el python y el math, tenemos una calculadora científica, y con el scikit-learn . . .

Mas información: help

Si queremos tener información sobre una librería, primero la importamos

```
>>> import math
```

Luego utilizamos el comando help().

```
>>> help(math)
```

Esto nos da información de toda la librería, si queremos consultar información de un comando:

```
>>> help(input)
```

```
>>> help(math.log)
```

Uso de librerías. os

Si queremos ejecutar algun comando de terminal shell o bash, esta la librería os.

```
>>> import os
>>> os.system('ls')
```

Si queremos chequear si existe un archivo: isfile

```
>>> import os
>>> os.path.isfile('prueba.tex')
```

Si queremos chequear si existe un directorio o un archivo: exists

```
>>> os.path.exists('prueba.tex')
```

Si queremos chequear si existe un directorio: isdir

```
>>> os.path.isdir('/home/programacion/readme')
```

Uso de librerías: datetime

Para manejo de fechas, python tiene la librería datetime

Si se necesita conocer la fecha de hoy:

```
>>> import datetime
>>> datetime.date.today()
```

Si se necesita introducir una fecha:

```
>>> date1=datetime.datetime(2007,1,1)
```

Para pasar una fecha a número de días: hoy.toordinal()

Para pasar un número de días a fechas: datetime.date.fromordinal(4535)