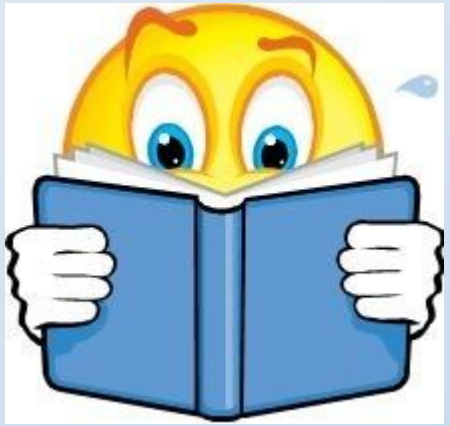
MÉTODOS NUMÉRICOS II

Licenciatura en informática
FACET-UNT
2017

Lic. Leonardo Albarracín
Septiembre 2017

CONCEPTOS BASICOS DE PYTHON



LENGUAJE COMPILADO

VS

LENGUAJE INTERPRETADO

LENGUAJE INTERPRETADO

- El código fuente escrito por un programador en un lenguaje de alto nivel, es **traducido por el interprete a un lenguaje entendible para la máquina**, instrucción por instrucción.

- **El proceso se repite cada vez que se ejecuta el programa el código.**

- Permiten el tipado dinámico de datos, es decir, **no es necesario inicializar una variable con determinado tipo de dato** sino que esta puede cambiar su tipo en condición al dato que almacene, entre otras características más.

LENGUAJE INTERPRETADO

También tienen por ventaja una **gran independencia de la plataforma donde se ejecutan**.

La principal desventaja de estos lenguajes es el tiempo que necesitan para ser interpretados. Al tener que ser traducido a lenguaje máquina con cada ejecución, este proceso es más lento que en los lenguajes compilados, sin embargo, algunos lenguajes poseen una máquina virtual que hace una traducción a lenguaje intermedio con lo cual el traducirlo a lenguaje de bajo nivel toma menos tiempo.

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

https://www.python.org

El intérprete estándar incluye un modo interactivo (intérprete de comandos)

- Las expresiones pueden ser introducidas una a una para ver el resultado de su evaluación inmediatamente.
- Posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa.

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

VARIABLES

- Se definen de forma dinámica: No se tiene que especificar cuál es su tipo de Antemano.

- Puede tomar distintos valores en otro momento, incluso de un tipo diferente al que tenía previamente.

- Se usa el símbolo **=** para asignar valores.

```
>>> x = 1

>>> x = "texto"    # Esto es posible porque los tipos son asignados dinámicamente
```

TIPOS DE DATOS

Números :
- -Integer
- -Float
- -Complejos

Cadenas

Tuplas

Listas

TIPOS DE DATOS

Integer

```
>>> 2+2
4
>>> 5*6
30
>>> 7/2 # La división entera redondea hacia abajo:
3
```

Float

```
>>> 3.0
```

Los operadores con tipos mixtos convierten el operando entero a coma flotante:

```
>>> 4 * 2.5 / 3.3
3.0303030303
>>> 7. / 2
3.5
```

TIPOS DE DATOS

Cadenas de caracteres o String

```
>>> 'Hola mundo‘
'Hola mundo‘
```

Tuplas

Una tupla consta de cierta cantidad de valores separada por comas, por ejemplo:

```
>>> t = 12345, 54321, '¡hola!‘
>>> t[0]
12345

>>> t
(12345, 54321, '¡hola!')

>>> u = t, (1, 2, 3, 4, 5) # Se pueden anidar tuplas:
>>> u
((12345, 54321, '¡hola!'), (1, 2, 3, 4, 5))
```

TIPOS DE DATOS

Listas

De los tipos de datos secuenciales que hemos visto el mas versátil es el de lista.

```
>>> a = ['fiambre', ‘quesos', 100, 1234]
>>> a
['fiambre', ‘quesos', 100, 1234]
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
[‘quesos', 100]
>>> a[:2] + [‘pan', 2*2]
['fiambre', ‘quesos', ‘pan', 4]
```

TIPOS DE DATOS

Listas

Para apilar un elemento, usa append(). Para recuperar el elemento superior de la pila, usa pop() sin un índice explícito.

Por ejemplo:

```
>>> pila = [3, 4, 5]
>>> pila.append(6)
>>> pila.append(7)
>>> pila
[3, 4, 5, 6, 7]
>>> pila.pop()
7
>>> pila
[3, 4, 5, 6]
>>> pila.pop()
6
>>> pila
[3, 4, 5]
```

SINTAXIS

Comentarios

detrás de **#**

Asignación

se usa **=**

Ej:

```
>>> # Esto es un comentario
>>> B=7 #asignación
```

SINTAXIS

Estructuras de control

IF

```
if condicion:
      acciones
elif condicion:
      acciones
else:
      acciones
```

WHILE

```
while condicion:
      acciones
```

And or not

True False

FOR

```
>>>for variable in elemento iterable (lista, cadena,
range, etc.):
            acciones
```

SINTAXIS

EJEMPLOS

```
numA = input("ingrese numero A: ")
numB = input("ingrese numero b: ")

if numA > numB:
    print "numero A es mayor"
elif numA < numB:
    print "numero B es mayor"
else:
    print "Ambos numeros son iguales"
```

SINTAXIS

EJEMPLOS

```
print "Comienzo"
for i in [0, 1, 2]:
    print "Hola. Ahora i vale", i
print "Fin"
```

```
print "Comienzo"
for i in [1, 1, 1, 1]:
    print "Hola. Ahora i vale", i
print "Final"
```

```
print "Comienzo"
for i in []:
    print "Hola. Ahora i vale", i
print "Final"
```

```
print "Comienzo"
for i in ["Alba", "Benito", 27]:
    print "Hola. Ahora i vale", i
print "Final"
```

```
i = 1
while i <= 50:
  print(i)
  i = 3*i + 1
print "Programa terminado"
```

LIBRERIAS

Python contiene una gran cantidad de módulos, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.

Entre los principales se destacan:

LIBRERIAS

NumPy: Es el paquete fundamental para el cálculo numérico. En él se definen los tipos de arrays y matrices numéricas y las operaciones básicas sobre ellos.

SciPy: Es una colección de algoritmos numéricos y herramientas para dominios específicos incluyendo el procesamiento de señales, la optimización de funciones, integración, resolución de ecuaciones diferenciales ordinarias, estadísticas, entre otras.

Matplotlib: Es una biblioteca para la generación de gráficos de calidad 2D y 3D a partir de los datos contenidos en listas o arrays.

LIBRERIAS

¿Cómo utilizamos los módulos?

Para poder utilizar estos módulos hay que importarlas

```
import numpy

import numpy as np

from numpy import *

From numpy import arange
```

LIBRERIAS

Numpy

identity(n,dtype). Devuelve la matriz identidad. *n* es el número de filas y columnas que tendrá la matriz y *dtype* es el tipo de dato. Este argumento es opcional. Si no se establece, se toma por defecto como flotante.

ones(shape,dtype). Crea un array de unos compuesto de *shape* elementos.

zeros(shape, dtype). Crea un array de ceros compuesto de “shape” elementos”.

arange([start,]stop[,step,],dtype=None). Crea un array con valores distanciados *step* entre el valor inicial *star* y el valor final *stop*. Si no se establece *step* python establecerá uno por defecto.

linspace(start,stop,num,endpoint=True,retstep=False). Crea un array con valor inicial *start*, valor final *stop* y *num* elementos.

LIBRERIAS

Matplotlib

```
import matplotlib.pyplot as plt
```

figure(num = None, figsize = (8, 6), dpi = 80, facecolor = 'w', edgecolor = 'k')
subplot(numRows, numCols, plotNum):Permite incluir varias gráficas en una única figura.
plot(x, y, linestyle, linewidth, marker): Permite incluir varias gráficas en una única figura. Tanto x como y pueden ser abcisas tuplas, listas o arrays. La única condición es que el tamaño de ambas debe ser el mismo ya que en caso contrario python nos devolverá un fallo de tipo dimesión.
show(): Presenta las figuras en pantalla.
xlabel('s', comandos_optativos): Etiqueta el eje X
ylabel('s', comandos_optativos): Etiqueta el eje Y
title('s', comandos_optativos): Coloca un titulo a la gráfica
axis() Establece u obtiene las propiedades de los ejes

LIBRERIAS

Math

```
import math as m
```

fabs(x): Return the absolute value of x.
factorial(x): Return x factorial. Raises ValueError if x is not integral or is negative.
exp(x): Return e**x.
log(x[, base]): With one argument, return the natural logarithm of x (to base e).

With two arguments, return the logarithm of x to the given base, calculated as log(x)/log(base).

log10(x): Return the base-10 logarithm of x. This is usually more accurate than log(x, 10).
sqrt(x): Return the square root of x.
cos(x): Return the cosine of x radians.
sin(x): Return the sine of x radians.
tan(x): Return the tangent of x radians.

FUNCIONES

```
def nombrefunción(arg1,arg2..):
    instrucción_1
    instrucción_2
    ..................

    instrucción_N
    return ....
```

Si no se usa **return**, la función devuelve None.

BIBLIOGRAFIA

- http://lsi.ugr.es/~pdo/Seminarios/Python2005.pdf
- http://wp.df.uba.ar/wtpc/wp-content/uploads/sites/6/2016/03/02_basico_python_jarne.pdf
- http://pendientedemigracion.ucm.es/info/aocg/python/modulos_cientificos/numpy/index.html
- http://pendientedemigracion.ucm.es/info/aocg/python/modulos_cientificos/matplotlib/index.html
- https://docs.python.org/2/library/math.html#math.sqrt