Introducción a Python

Comandos básicos y gráficas en Python

Luis Alejandro Morales, Ph.D.
lmoralesm@unal.edu.co

Facultad de Ingeniería
Universidad Nacional de Colombia

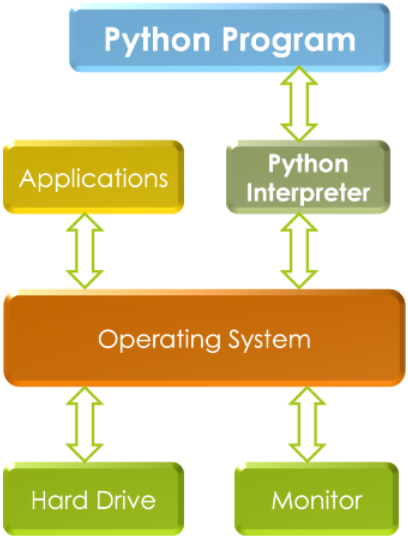
April 4, 2024

Outline

¿Que es Python?

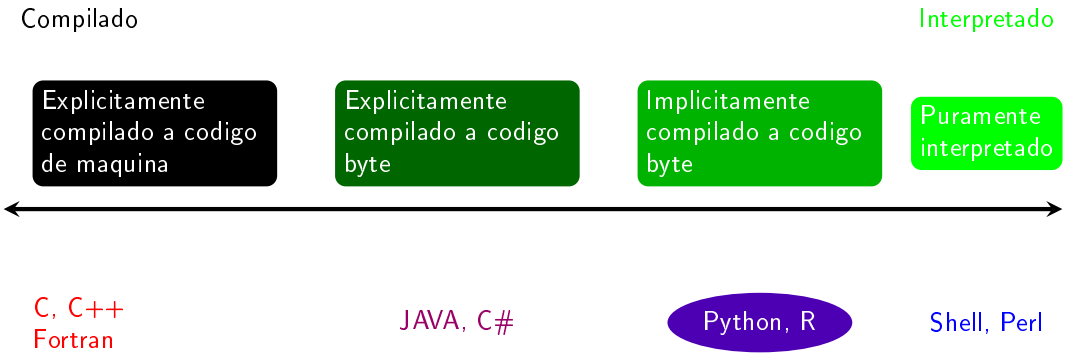
Python es un lenguaje de programación interpretado, orientado a objetos, de alto nivel y con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con escritura dinámica y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de secuencias de comandos o para conectar componentes existentes. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo de mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en formato fuente o binario sin costo para todas las plataformas principales y se pueden distribuir gratuitamente.

¿Como se ejecuta python en su computador?



- Python programs no se ejecutan directamente en el sistema opertativo (OS).
- Otro programa llamado interprete o máquina virtual toma el programa y lo pasa (ejecución) a lenguage de maquina (entendido por el OS).
- Aqui se escribiran programas ejecutados por el interprete.

¿Que tipo de lenguage es Python?



¿Para que sirve Python y sus ventajas?

Untilidades de Python

- Science: Deterministic and statistical modelling
- Instrumental control
- Embedded systems
- Web services
- On-line games

Ventajas de Python

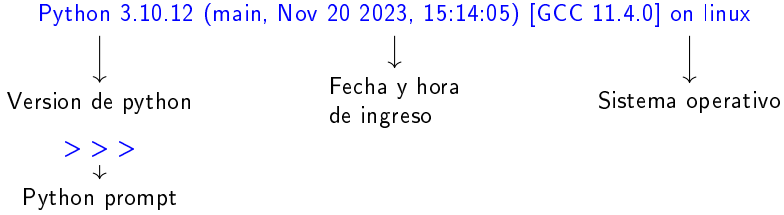
- Relativamente facil de aprender
- Gran numero de librerias
- Versatil e independiente de la plataforma
- Comunidad con fuerte soporte

¿Como instalar Python?

- Python tiene dos versiones:
  - 2.x: Version más antigua, carece de mantenimiento.
  - 3.x; Version reciente, con suporte actual.
- Minimo hardware requerido: $>$ 4GB de RAM, >5 GB de disco duro libre (para la instalación de librerias se requier más espacio)
- Unix/Linux y Mac OS, Python viene instalado por defecto,:)
- MS Windows
  - Descargar el installador de Python en (https://www.python.org/downloads/windows/)
  - Ejecutar el installador `pytho3.x.x.exe`
  - Personalizar la instalación
  - Installar Python
  - Verificar la installación: Busque `cmd` y escriba `python -version`

¿Como usar Python?

1. Ejecutar comandos y codigos directamente desde el interprete de Python
   1.1 Abra la termninal (busque `cmd`)
   1.2 Ejecute `python3` o `python`. Producirá:



2. Correr programas escritos en un archivo (e.g. `myprogram.py`)
   2.1 Abra la termninal (busque `cmd`)
   2.2 Para ejecutar el programa: `python3 myprogram.py` o `python myprogram.py`
3. Salir de Python:
   3.1 >>> `exit()`
   3.2 >>> `quit()`
   3.3 Ctrl + D

¿Como ejecutar un programa de Python?

1. Abra un archivo nuevo en su editor de texto (e.g. `vi/vim`, `emacs`, `notepad`)
2. Nombre su archivo como e.g. `myprogram.py`
3. Dentro de ese archivo escribir:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# Mi primer programa
print('Hello, world!')
```

4. En la consola de comandos, asegurese que usted esta en el directorio que contiene `myprogram.py`.
5. Ejecutar el programa como: `python3 myprogram.py` o `python myprogram.py`
6. Note que el `myprogram.py` puede leer parametros de entrada: `python3 myprogram.py para1 para2 ··· paran`

Outline

Primer comando en Python

```
>>> print('Hello, world!')
```

Python command

```
Hello, world!
```

Command output

Tipos de datos y operaciones en Python

- integers → $4\ 15110\ -53\ 0$
- Floating point numbers → $4.0\ 0.80333333333\ 7.34e+014$
- Strings → "hello" "A" " " "" 'there' '"' '15110'
- Booleans → `True` `False`
- Literal none → `None`

El interprete de python es como una calculadora:

```
>>> 2+3*5
    17
```

Orden de evaluación de los operadores

() ⟹ ** ⟹ * / % ⟹ + -

- + Addition
- − Substraction
- ∗ Multiplication
- / Division
- // Integer division
- ∗∗ Exponentiation
- % Modulo (remainder)

Operaciones logicas

Matemáticas	Python
$=$	$==$
$\neq$	$!=$
$<$	$<$
$>$	$>$
$\leq$	$<=$
$\geq$	$>=$

```
>>> a = 5
>>> 5 < a < 10
True
```

$=$	varname $=$ value	Asginacion de value a varname
$==$	$\text{varname}_1 = \text{varname}_2$	Comparación si varname_1 es igual a varname_2

Operadores boleanos: and, or y not

```
>>> 4 < 5 and 6 < 7
True
>>> 4 < 5 and 6 > 7
False
>>> 4 < 5 or 6 < 7
True
>>> 4 < 5 or 6 > 7
True
>>> not 6 < 7
False
>>> not 6 > 7
True
```

Expresiones y declaraciones

- Python evalua una expresion para obtener un resultado, e.g.:

```
>>> 2.5 + sin(30)
```

```
3
```

- Python ejecuta una declaración para realizar una acción que tiene un efecto, e.g.:

```
>>> print('Hello, world!')
```

```
Hello, world
```

Variables

- Es un lugar en la memoria donde se puede almacenar informacion.
- En Python, se almacena un valor en memoria a travéz de una declaración de asignación.
- Una variable puede contener cualquier tipo de valor u objeto.

```
>>> a = 5          Declaración de asignación
>>> a              Expresión
5                  Output
```

Note que a es una variable tipo integer almacenada en la memoria. a puede ser utilizada como:

```
>>> b = 2*a        Declaración de asignación
>>> b              Expresión
10                 Output
```

Una variable a puede actualizarse:

```
>>> a = 5
>>> a = a + 5
>>> a
10
```

Nombre de las variables

- Todas las variables deben comenzar con una letra. Es recomendado iniciar con una letra minuscula.
- Las siguientes letras pueden ser mayusculas, minusculas o números.
- Los nombres de las variables son sensibles a las mayusculas y minusculas:

$$\text{var} \neq \text{Var}$$

$$\text{a2} \neq \text{A2}$$

Abreviaciones sintacticas

$$
\begin{aligned}
\text{varname} += 10 &\equiv \text{varname} = \text{varname} + 10 \\
\text{varname} -= 10 &\equiv \text{varname} = \text{varname} - 10 \\
\text{varname} *= 10 &\equiv \text{varname} = \text{varname} * 10 \\
\text{varname} /= 10 &\equiv \text{varname} = \text{varname} / 10 \\
\text{varname} **= 10 &\equiv \text{varname} = \text{varname} ** 10 \\
\text{varname} \%= 10 &\equiv \text{varname} = \text{varname} \% 10
\end{aligned}
$$

Estructuras de datos en Python

`list`: Una lista es una colección de datos

- Del mismo o de diferente tipo

```
>>> l1 = [1, 2, 4]
>>> l2 = [1.0, 'abc', 23, True]
```

- Pueden incluir listas dentro de listas

```
>>> l = [[1, 2, 4], [2.5, None],['ab1', 'ac2']]
```

- Una lista vacia se expresa como:

```
>>> l = []
```

- Se pueden acceder a los elementos de una lista a través de indices

```
>>> l = [1, 2, 4, 10, 23]
>>> l[0]
1
>>> l[1:2]
2 4
```

`count()`	Calcula el numero de elementos de una lista.
`append()`	Adiciona un elemento al final de la lista.
`clear()`	Remueve todos los elementos de una lista.
`sort()`	Ordena los elementos de la list.
`insert()`	Insert elementos en una posición determinada.

Estructuras de datos en Python

dictionary: Es una colección de elementos, donde cada uno tiene una llave key y un valor value. Los diccionarios se pueden crear con paréntesis {} separando con una coma cada par key: value. En el siguiente ejemplo tenemos tres keys que son el nombre, la edad y el documento.

```python
d1 = {
  "Nombre": "Sara",
  "Edad": 27,
  "Documento": 1003882
}
print(d1)
#{'Nombre': 'Sara', 'Edad': 27, 'Documento': 1003882}
```

```python
d2 = dict([
       ('Nombre', 'Sara'),
       ('Edad', 27),
       ('Documento', 1003882),
])
print(d2)
#{'Nombre': 'Sara', 'Edad': '27', 'Documento': '1003882'}
```

```python
d3 = dict(Nombre='Sara',
          Edad=27,
          Documento=1003882)
print(d3)
#{'Nombre': 'Sara', 'Edad': 27, 'Documento': 1003882}
```

Diccionario anidado

Estructuras de datos en Python

```python
print(d1['Nombre'])      # Sara
print(d1.get('Nombre')) # Sara

d1['Nombre'] = "Laura"
print(d1)
#{'Nombre': Laura', 'Edad': 27, 'Documento': 1003882}
```

Acceder y modificar elementos

```python
# Imprime los key y value del diccionario
for x, y in d1.items():
    print(x, y)
#Nombre Laura
#Edad 27
#Documento 1003882
#Direccion Calle 123
```

Iterar diccionario

```python
anidado1 = {"a": 1, "b": 2}
anidado2 = {"a": 1, "b": 2}
d = {
  "anidado1" : anidado1,
  "anidado2" : anidado2
}
print(d)
#{'anidado1': {'a': 1, 'b': 2}, 'anidado2': {'a': 1, 'b': 2}}
```

Diccionario anidado

- `clear()`: Elimina todo el contenido del diccionario.
- `get(<key>[,<default>])`: Este método nos permite consultar el `value` para un `key` determinado.
- `items()`: Devuelve una lista con los keys y values del diccionario.
- `keys()`: Devuelve una lista con todas las keys del diccionario.
- `values()`: Devuelve una lista con todos los values del diccionario.

Otras estructuras de datos en Python

- `tuple`: Es una colección de objetos o variables similar a una `list` en donde sus objetos son inmutables. Esto quiere decir que los objetos en un `tuple` no puden ser creados o removidos una vez creado el `tuple`. E.g. `Tuple = ('Geeks', 'For')`.
- `set`: Es una coleccion de datos en python la cual mutable similar a una lista y no permite duplicados de datos. E.g. `Set = set([1, 2, 'Geeks', 4, 'For', 6, 'Geeks'])`

Condicionales en Python

`if () elif: else:` son usadas para ejecutar expresiones dependiendo de una condicion logica

```
a = 33
b = 200
if b > a:
  print("b is greater
    than a")
```

producira: b is greater than a

```
a = 33
b = 33
if b > a:
  print("b is greater
    than a")
elif a == b:
  print("a and b are
    equal")
```

producira: a and b are equal

```
a = 200
b = 33
if b > a:
  print("b is greater
    than a")
elif a == b:
  print("a and b are
    equal")
else:
  print("a is greater
    than b")
```

producira: a is greater than b

Formas compactas de escribir condicionales:

```
a = 200
b = 30
if a > b: print("a is greater than b")
```

producira: a is greater than b

```
a = 330
b = 330
print("A") if a > b else print("=")
```

producira: =

Condicionales en Python

Uso de operadores logicos and, or y not en condicionales

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions
    are True")
```

producira: Both conditions are True

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of
    the conditions is
    True")
```

producira: At least one of the conditions is True

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater
    than b")
```

producira: a is NOT greater than b

```
x = 41
if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Condicional anidado

producira: Above ten, y and also above

```
a = 33
b = 200
if b > a:
  pass
```

uso de pass

Los for loops

```
fruits = ["apple", "
    banana", "cherry"]
for x in fruits:
  print(x)
```

Loop a través de una lista

producirá:

apple
banana
cherry

```
for x in "banana":
  print(x)
```

Loop a través de un string

producirá:

b
a
n
a
n
a

```
fruits = ["apple", "
    banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Uso de break

producirá:

apple

Los for loops

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Uso de `continue`

producirá:

```
apple
cherry
```

```
for x in range(6):
  print(x)
```

La función `range()`

producirá:

```
0
1
.
.
.
5
```

```
for x in range(2, 30, 3):
  print(x)
```

La función `range()`

producirá:

```
2
5
.
.
.
29
```

Los `while` loops

```
i = 1
while i < 6:
  print(i)
  i += 1
```

producirá:

1
2
3
4
5

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Uso de `break`

producirá:

1
2
3

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Uso de `continue`

producirá:

1
2
4
5
6

Ejercicio

Calculo de la profundidad normal (y_n) en canales

Escribir un programa en Python para determinar la profundidad normal (y_n) en un canal trapezoidal cuya base (b) es 10 m, la pendiente longitudinal del canal (S_o) es 0.001, la pendiente lateral (z) es 2, el factor de rugosidad de Manning (n) es 0.013 y el caudal transportado (Q) es 30 m^3/s, utilizando la *ecuación de Manning*:

$$Q = \frac{1}{n} A R^{2/3} S_o^{1/2}$$

Utilize el método de *Newton-Raphson*:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

para resolver la ecución de Manning.

Leer datos externos

Imprimir datos en Python

Outline

Paquetes

Graficar datos

Atributos de las graficas

Multiples axes en una grafica