Introducción Rápida a Python

Carlos E. Alvarez[1].

[1]Dep. de Matemáticas, Universidad del Rosario

Bogotá, Octubre 2016

Contenido

Características del lenguaje Python

- Lenguaje de *scripting*. Pensado para desarrollar rápidamente
- Lenguaje interpretado
- Uso de recolección de basura
- Tipado débil

Contenido

Tipado dinámico

- **Creación de variables:** Se crean cuando el código les asigna (=) un valor
- **Tipo de una variable:** Una variable no tiene tipo, se refiere a un objeto (que sí tiene tipo)
- **Uso de una variable:** Cuando aparece en una expresión, es reemplazada por el objeto referido

Tipos numéricos

- `int` (8, 16, 32 y 64 bits)
- `float` (16, 32 y 64 bits)
- `long` (precisión *infinita*)
- `complex` (2 x 32 y 2 x 64 bits)

Operaciones numéricas: $+$, $-$, $*$, $/$, $\%$, $**$.
Operaciones booleanas: $==$, $!=$, $>$, $>=$, $<$, $<=$, `is`, `is not`.

Operaciones en cadena: $z < y < x \rightarrow z < y$ **and** $y < x$

Booleanos

`bool` (`True` o `False`)

Operaciones: `and`, `or`, `not`, $\wedge$(xor).

Cadenas de caracteres

Tipo `str`

- `str`: Texto Unicode
- `bytearray`: Datos binarios

Declaración:

`S='Hola mundo!'` o `S="Hola mundo!"`

Cadenas de caracteres

Manipulación:

- Concatenación: `S1+S2`
- Repetición: `n * S1`
- Indexar: `S[i]` se refiere al i-ésimo caracter
- Secciones: `S[i:j]` se refiere a la subcadena de la posición i a la posición j-1

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Cadenas de caracteres

Métodos y funciones comunes:

- Longitud: `len(S)`
- Busqueda de subcadena: `S.find(sub,ini,fin)`
- Dividir usando un delimitador: `S.split(',')`
- Remover salto de línea final: `S.rstrip('\n')`

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Cadenas de caracteres

Ejercicios:

1. Declare las cadenas `'Hola'` y `'Mundo'` y concatenelas para imprimir,usando la función `print()`, la frase `'Hola Mundo'`
2. Declare la cadena de caracteres `cad='Hola Mundo!'` e imprima la subcadena `'Mundo'`
3. Use la función `input()` para pedir al usuario una cadena de caracteres. Busque la subcadena `'abc'` y, de encontrarla, imprima la posición en la que se encuentra por primera vez.

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Listas

Tipo `list`

- Colección ordenada de objetos arbitrarios
- Longitud variable, anidadación
- Mutables
- Arreglos de referencias a objetos

Declaración:

```
L=[3,'abc',45.2,['py','thon']]
```

Acceso:

`L[2]` $\rightarrow$ `45.2`

`L[3][0]` $\rightarrow$ `'py'`

`L[1:3]` $\rightarrow$ `['abc,45.2']`

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Listas

Métodos y funciones comunes:

- Longitud: `len(L)`
- Añadir elementos: `L.append(a)`
- Organizar (alfabético/numérico): `L.sort()`
- Crear una lista con una secuencia dada de enteros: `range(ini,fin,step)`

Listas

Ejercicios:

1. Cree una lista con los números del 1 al 9. obtenga la sección de los números del 3 al 5
2. Usando listas, cree un arreglo de 3x3 que contenga los números del 1 al 9. Imprima el elemento en la segunda columna, primera fila
3. Cree dos listas y concaténelas usando el operador +

Diccionarios

Tipo `dict`

- Colección *no* ordenada de objetos arbitrarios
- Acceso por clave
- Longitud variable, anidación
- Mutables
- Sus elementosson pares `clave:valor`

Declaración:

`D={'nombre':'Juan', 'edad':45, 'tel':3124567}`

Acceso:

`D['edad']` $\rightarrow$ `45`

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Tuplas

Tipo `tuple`

- Colección ordenada de objetos arbitrarios
- Longitud fija, anidación
- Inmutables

Declaración:

```
T=(1,'abc',2,3)
```

Acceso:

`T[2]` $\rightarrow$ `2`

Copia de una estructura de datos

Si definimos una estructura, por ejemplo:

```
>>> l1 = [1,2,3,4]
```

y queremos hacer una copia `l2` **¡escribir `l2=l1` no funciona!** por que se refiere al mismo espacio en memoria.

Se deben copiar los elementos explícitamente, por ej.:

```
>>> l2 = l1[:]
```

Archivos

Tipo `file`

Métodos y funciones comunes:

- Abrir un archivo: `open(nombre,modo)`
- Leer completamente a una `str`: `F.read()`
- Leer una línea a una `str`: `F.readline()`
- Leer todas las líneas a una lista de `str`: `F.readlines()`
- Escribir una `str` a un archivo: `F.write(cadena)`
- Cerrar un archivo: `F.close()`

Archivos

Ejemplo de escritura:

```
>>> arch1 = open('archivo.txt', 'w')
>>> arch1.write('Esta es una prueba')
>>> arch1.close()
```

Ejemplo de lectura:

```
>>> arch2 = open('archivo.txt', 'r')
>>> texto = arch2.read()
>>> arch2.close()
>>> print(texto)
```

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Contenido

Universidad del Rosario

MACC
Matemáticas Aplicadas y Ciencias de la Computación

Estructura

1. Un programa se compone de módulos
2. Un módulo contiene enunciados
3. Un enunciado contiene expresiones
4. Una expresión crea y procesa objetos

Enunciados

- Asignación: `a, b = 6, 'a'`
- Condicionales: `if/elif/else`
- Ciclos: `for, while`
- Definición de funciones/métodos: `def`
- etc.

Asignación

1. Se crean referencias a objetos
2. Nombre se crea al momento de la primera asignación
3. Algunas operaciones realizan asignaciones implícitamente

Operaciones de asignación:

- Básica: `a = 'abc'`
- Secuencia: `a, b = 1, 2` o `a, b, c = '123'`
- Múltiple objetivo: `a = b = 8`
- Aumentado: `a += 6`

Operaciones de Impresión

Función `print()` (python 3.X):
`print(objetos, separador, fin, archivo)`

- Objetos: Objetos a imprimir (separados por coma)
- Separador: Cadena de caracteres que aparece impresa entre los objetos (`' '` por defecto)
- Fin: Última cadena de caracteres (`'\n'` por defecto)
- Archivo: Objeto tipo `file` a donde se imprime

Operaciones de Impresión

Ejemplos (python 3.X):

```
>>> x = 'hola'
>>> y = 50
>>> z = 'amigos'
>>> print(x,y,z)
hola 50 amigos
>>> print(x, y, z, sep=',', end='...')
hola,50,amigos...
>>> print(x+' '+str(y)+' '+z)
hola 50 amigos
```

Enunciados Condicionales

Forma general:

```
if prueba1:
    enunciado1
elif prueba2:
    enunciado2
else:
    enunciado3
```

Enunciados Condicionales

Ejemplo:

```
>>> eleccion = 'jamon'
>>> if eleccion == 'huevos':
...     print(1.0)
... elif eleccion == 'pan':
...     print(2.0)
... elif eleccion == 'jamon':
...     print(3.0)
... else:
...     print('Mala eleccion')
3.0
```

Enunciados Condicionales

'Switch' usando un diccionario:

```
>>> eleccion = 'jamon'
>>> dic = {'huevos': 1.0,
...        'pan': 2.0,
...        'jamon': 3.0}
>>> print(dic.get(eleccion, 'Mala eleccion')
3.0
```

Aquí, el método `get` retorna el valor relacionado a `eleccion` si esta se encuentra entre las claves del diccionario. En caso contrario retorma por defecto el segundo algumento `'Mala eleccion'`.

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Enunciados Condicionales

Ejercicios:

1. Escriba un programa que pida dos números y diga cual es el mayor y cual es el menos, o diga si son iguales en caso que lo sean.
2. Escriba un programa que pida dos enteros y diga si el mayor es múltiplo del menor.
3. Escriba un programa que reciba los dos coeficientes de una ecuación lineal y dé la solución.

Ciclos

Ciclos `while`:

`while` *prueba*:
    enunciado

Ejemplo:

```
>>> x = 'hola'
>>> while x:
...     print(x)
...     x = x[1:]
hola
ola
la
a
```

Ciclos

`break`: Sale del ciclo.

```
>>> a = 0
>>> while True:
...   print(a)
...   a += 1
...   if a > 3: break
0
1
2
3
```

Ciclos

`continue`: Salta al inicio del ciclo.

```
>>> a = 0
>>> while a < 6:
...   a += 1
...   if 2 < a < 5: continue
...   print(a)
1
2
5
6
```

Ciclos

Ciclos `for`:

`for` *objetivo* `in` *objeto*:
    enunciado

Donde el *objeto* es un iterable.

Ejemplo:

```
>>> x = 'hola'
>>> for i in x:
...   print(x)
h
o
l
a
```

Ciclos

Función `range()`:

`range(`*ini*, *fin*, *paso*`)`

Retorna una lista de valores desde *ini* hasta *fin-1* saltando de *paso*.

```
>>> for i in range(2,9,2):
...   print(i)
2
4
6
8
```

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Comprensión de listas

Se ueden incluir ciclos y enunciados condicionales en la declaración de una lista.

`mi_lista = [`*func*`(i) for i in` *objeto* `if` *condición*`]`

Ejemplo:

```
>>> l = [1,2,'a',5,4.6]
>>> l1 = [i**2+1 for i in l if type(i)==int]
>>> l1
[2, 5, 26]
```

Ciclos

Ejercicios:

1. Escriba un programa que pida un número entero y sigue pidiendolo mientras se le ingresen enteros. Una vez se le ingresa algo diferente a un entero se detiene.
2. Escriba un programa que pida un entero mayor que 0 y a continuación pida esa misma cantidad de números enteros.
3. Escriba un programa que pida dos enteros y calcule la suma de todos los enteros desde el mínimo de los dos números hasta el máximo.

Contenido

Universidad del Rosario

MACC
Matemáticas Aplicadas y Ciencias de la Computación

Funciones

Declaración:

def *nombre*(*arg1*, *arg2*, . . . , *argN*) :
    enunciados
    `return` *valor*

Ej:

```
>>> def minimo(a, b):
...   res = a
...   if a > b:
...     res = b
...   return res
>>>
>>> minimo(3,6)
3
```

Funciones

`def` ejecuta en el *runtime*.

```
>>> from random import random
>>> a = random()
>>> if a > 0.5: ...   def func():
...       print('numero mayor que 0.5')
>>> else:
...   def func():
...       print('numero menor o igual a 0.5')
>>>
>>> func()
```

MACC
Matemáticas Aplicadas y
Ciencias de la Computación

Funciones

Las variables declaradas dentro de una función son locales.

Los intérpretes de python no siempre cumplen con esto!

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Funciones

Ejercicios:

1. Escriba una función que reciba una cadena de caracteres, que puede consistir en números y letras, filtre las letras y retorne solamente los números contenidos en dicha cadena. Por ejemplo, si se ingresa `gato123felino45`, debe retornar `12345`.
2. Escriba una función que reciba una cadena de caracteres y cuente con que frecuencia se repite cada uno de los caracteres contenidos en la cadena, sin tener en cuenta si las letras son mayusculas o minusculas. La función debe imprimir cada uno de los caracteres contenidos, sin repetir y con su frecuencia respectiva.

Contenido

Módulos

Comandos:

`import`: Hace que un importador cargue un módulo.

`from`: Permite que el importador tome nómbres específicos de un módulo.

Ejs:

```
>>> from math import sqrt
>>> sqrt(4.)
2.0
```

o

```
>>> import math
>>> math.sqrt(4.)
2.0
```

Módulos

Ejemplo de creación de un módulo:

Archivo funcion.py:

```
def promedio(a, b):
    prom = (a + b)/2
    return prom
```

Script que carga el módulo:

```
>>> import funcion
>>> funcion.promedio(3.,8.)
5.5
```

Módulos

Ejercicio:

1. Cree un módulo que incluya las siguientes funciones:
   a) `area`: Recibe el radio de un círculo y retorna su área.
   b) `retirar`: Recibe una cadena de caracteres y un caracter. La función retorna una cadena que es la original pero a la que se le han retirado todas las ocurrencias del caracter recibido.
   c) `primo`: Recibe un entero y retorna True si el numero es primo o False en caso contrario.

   Cargue su módulo en un programa y utilice las funciones.

Contenido

MACC
Matemáticas Aplicadas y
Ciencias de la Computación

Clases

Definición de nuevos tipos de objetos.

- Multiples instancias
- Herencia
- Sobrecarga de operadores

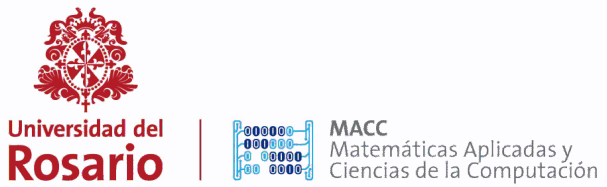
Clases

Ejemplo:

```
>>> class MiClase:
      def inidatos(self, valor):
         self.datos = valor
      def mostrar(self):
         print(self.datos)
```

Creación de instancias:

```
>>> x = MiClase()
>>> y = MiClase()
>>> x.inidatos('Hola!')
>>> y.inidatos(1.2435)
```



Clases

En python se permite cambiar atributos e inclusive generar nuevos atributos desde fuera de la clase:

```
>>> x.datos = 'nuevo'
>>> x.mostrar()
nuevo
>>> x.nuevo_atrib = 10
```

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación

Herencia

Definamos otra clase:

```
>>> class OtraClase(MiClase):
      def mostrar(self):
        print('Valor actual = "%s" ' % self.datos)
```

`OtraClase` hereda `inidatos` de `MiClase` y modifica `mostrar`

Clases

Ejercicios:

1. Cree la clase `circulo`, la cual tiene como atributos su radioy la posición de su centro en coord. cartesianas y como métodos `area`, que calcula y retorna su área.
2. Cree la clase `empleado`, que tiene como atributos su número de identificación, nombre, apellido y salario. Como métodos tiene `salario_anual` que retorna el salario anual y `aumento_salarial` que recive un porcentaje y calcula el salario aumentado por dicho porcentaje.