44-PROGR-mensajes-error

October 12, 2017

Errores tontos

```
In [1]: L = []
        for n in srange(7)
            L.append(n^2)
            print L


          File "<ipython-input-1-d6f9d662070c>", line 2
        for n in srange(Integer(7))
                                   ^
    SyntaxError: invalid syntax
```

```
In [2]: L = []
         for n in srange(7):
            L.append(n^2)
        print L


          File "<ipython-input-2-9b120a3edc8d>", line 2
        for n in srange(Integer(7)):
        ^
    IndentationError: unexpected indent
```

```
In [3]: L = []
        for n in srange(7):
            L.append(n^2)
        print L
```

```
[0, 1, 4, 9, 16, 25, 36]
```

```
In [4]: L = []
        for n in srange(7):
            L.append(n^2
        print L
```

```
  File "<ipython-input-4-f08865bdb0b9>", line 4
    print L
        ^
SyntaxError: invalid syntax
```

```
In [5]: def cuadrados(n):
            L = []
            for j in srange(j):
                L.append(n^2)
            return L
        cuadrados(5)
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)

<ipython-input-5-9d26b5f03370> in <module>()
  4         L.append(n**Integer(2))
  5     return L
----> 6 cuadrados(Integer(5))


<ipython-input-5-9d26b5f03370> in cuadrados(n)
  1 def cuadrados(n):
  2     L = []
----> 3     for j in srange(j):
  4         L.append(n**Integer(2))
  5     return L


UnboundLocalError: local variable 'j' referenced before assignment
```

Sage no sabe el valor de la variable *j* porque no se lo hemos dicho.

```
In [6]: srange(10)[10]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)

<ipython-input-6-33c4e8ea890b> in <module>()
----> 1 srange(Integer(10))[Integer(10)]
```

```
IndexError: list index out of range
```

La lista *srange*(10) tiene 10 elementos pero sus índices van del cero al nueve. Este error 'list index out of range' se produce frecuentemente después de manipular (cortar, concatenar, etc.) listas, y se suele resolver mediante 'prueba y error'.

```
In [7]: def recursion(N):
            if N == 1:
               return [1]
            return [N]+recursion(N-1)

In [8]: recursion(3)

Out[8]: [3, 2, 1]

In [9]: L = recursion(995)

        ---------------------------------------------------------------------------

        RuntimeError                              Traceback (most recent call last)

        <ipython-input-9-b84716e49006> in <module>()
    ----> 1 L = recursion(Integer(995))


        <ipython-input-7-6d001f74d241> in recursion(N)
          2     if N == Integer(1):
          3        return [Integer(1)]
    ----> 4     return [N]+recursion(N-Integer(1))


        ... last 1 frames repeated, from the frame below ...


        <ipython-input-7-6d001f74d241> in recursion(N)
          2     if N == Integer(1):
          3        return [Integer(1)]
    ----> 4     return [N]+recursion(N-Integer(1))


        RuntimeError: maximum recursion depth exceeded while calling a Python object

In [10]: L = recursion(994)

         ---------------------------------------------------------------------------
```

```
RuntimeError                                Traceback (most recent call last)

<ipython-input-10-9cca3c3e60ca> in <module>()
----> 1 L = recursion(Integer(994))


<ipython-input-7-6d001f74d241> in recursion(N)
  2     if N == Integer(1):
  3        return [Integer(1)]
----> 4     return [N]+recursion(N-Integer(1))


... last 1 frames repeated, from the frame below ...


<ipython-input-7-6d001f74d241> in recursion(N)
  2     if N == Integer(1):
  3        return [Integer(1)]
----> 4     return [N]+recursion(N-Integer(1))


RuntimeError: maximum recursion depth exceeded while calling a Python object
```

Un error más sutil

```
In [11]: print n
         L = []
         for n in srange(n):
             L.append(n^2)
         print L
```

```
6
[0, 1, 4, 9, 16, 25]
```

Este código es incorrecto, pero produce una respuesta porque la variable n tiene un valor asignado previamente en la hoja, como vemos con la primera línea de la celda. Ese valor asignado viene de la segunda línea de la celda anterior, y el valor final de n es 6. Si cerramos la hoja y ejecutamos la siguiente celda la primera aparece el error.

No se puede llamar igual al límite del bucle y a la variable que se incrementa dentro de él.

```
In [12]: L = []
         for n in srange(n):
             L.append(n^2)
         print L
```

```
[0, 1, 4, 9, 16]
```

Las variables que aparecen dentro de la definición de una función de Sage son variables locales, es decir adquieren valores únicamente en la forma en que se los asigna la función, siendo irrelevantes los valores que la misma variable puediera tener previamente en la hoja:

```
In [13]: n = 5
         def funcion():
             n = 8
             print n
         funcion() #Valor que n adquiere dentro de la funcion
         print n #Valor de n fuera de la funcion
```

```
8
5
```

La función protege los valores de sus variables locales, y por eso se llaman locales.

```
In [14]: def cuadrados(n):
             L = []
             for n in srange(n):
                 L.append(n^2)
             return L
         cuadrados(5)
```

```
Out[14]: [0, 1, 4, 9, 16]
```

Sin embargo, en esta última celda no se produce error porque al llamar a la función *cuadrados* estamos asignando a n el valor 5, que es el que tiene cuando empieza el bucle, y entonces a Sage (Python) no le importa que, una vez que ha formado el $srange(5)$ la variable sobre la que se itera, la que se incrementa en cada vuelta del bucle, se vuelva a llamar n. De todas formas, es una gran fuente de posible confusión usar los mismos nombres para distintas variables y no debe hacerse nunca.

Enteros de Sage y enteros de Python

```
In [15]: [n.factor() for n in range(2,10)]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)

<ipython-input-15-6038e5741e35> in <module>()
----> 1 [n.factor() for n in range(Integer(2),Integer(10))]


AttributeError: 'int' object has no attribute 'factor'
```

```
In [16]: [n.factor() for n in srange(2,10)]
```

```
Out[16]: [2, 3, 2^2, 5, 2 * 3, 7, 2^3, 3^2]

In [17]: [n.digits() for n in range(2,10)]

        ---------------------------------------------------------------------------
        AttributeError                            Traceback (most recent call last)

        <ipython-input-17-c9be42a9f4ad> in <module>()
    ----> 1 [n.digits() for n in range(Integer(2),Integer(10))]


        AttributeError: 'int' object has no attribute 'digits'


In [18]: [n.factor() for n in srange(2,10)]

Out[18]: [2, 3, 2^2, 5, 2 * 3, 7, 2^3, 3^2]
```

Iterables

```
In [19]: def sumando(n):
             suma = 0
             for j in n:
                 suma += j
             return suma

In [20]: sumando(7)

        ---------------------------------------------------------------------------
        TypeError                                 Traceback (most recent call last)

        <ipython-input-20-401e252f197f> in <module>()
    ----> 1 sumando(Integer(7))


        <ipython-input-19-bbdf46a1eb55> in sumando(n)
          1 def sumando(n):
          2     suma = Integer(0)
    ----> 3     for j in n:
          4         suma += j
          5     return suma


        TypeError: 'sage.rings.integer.Integer' object is not iterable
```

```
In [21]: def sumando_c(n):
             suma = 0
             for j in srange(n):
                 suma += j
             return suma

In [22]: sumando_c(7)

Out[22]: 21
```

Un bucle *for* siempre debe recorrer los elementos de un contenedor iterable (lista, tupla, conjunto, cadena de caracteres o claves de un diccionario).

Errores de tipo

Cada función o método se puede aplicar a objetos de ciertos tipos, y una de las grandes ventajas de Python es que no es necesario declarar esos tipos a priori. Por ejemplo, los métodos para números enteros, en general, no se pueden aplicar a racionales o decimales, pero los de racionales o decimales sí se pueden aplicar a enteros.

```
In [23]: factor(3.3)

Out[23]: 3.30000000000000

In [24]: is_prime(3.3)

Out[24]: False

In [25]: (33).n()

Out[25]: 33.0000000000000

In [26]: ZZ(3.3)


    ---------------------------------------------------------------------------
    TypeError                                 Traceback (most recent call last)

    <ipython-input-26-bc176e31420e> in <module>()
    ----> 1 ZZ(RealNumber('3.3'))


    /usr/local/SageMath/src/sage/structure/parent.pyx in sage.structure.parent.Parent.__cal
    932         if mor is not None:
    933             if no_extra_args:
    --> 934                 return mor._call_(x)
    935             else:
    936                 return mor._call_with_args(x, args, kwds)


    /usr/local/SageMath/src/sage/structure/coerce_maps.pyx in sage.structure.coerce_maps.Na
```

```
    280             raise TypeError("Cannot coerce {} to {}".format(x, C))
    281         cdef Map m
--> 282         cdef Element e = method(C)
    283         if e is None:
    284             raise RuntimeError("BUG in coercion model: {} method of {} returned No
```

```
    /usr/local/SageMath/src/sage/rings/real_mpfr.pyx in sage.rings.real_mpfr.RealNumber._i
   2049             return n
   2050
-> 2051         raise TypeError("Attempt to coerce non-integral RealNumber to Integer")
   2052
   2053     def integer_part(self):
```

```
    TypeError: Attempt to coerce non-integral RealNumber to Integer
```

Este error nos dice que los números decimales no están contenidos en los enteros, mientras que la celda siguiente no da error porque los racionales se pueden expresar como decimales, aunque hay que tener en cuenta que los decimales tienen una precisión prefijada y se puede producir un redondeo.

```
In [27]: RR(3/4);RR(12345678912345678/10^20)
```

```
Out[27]: 0.000123456789123457
```

Límites (overflow)
Con enteros:

```
In [28]: nth_prime(10^32+1)
```

```
    ---------------------------------------------------------------------------
    OverflowError                             Traceback (most recent call last)

    <ipython-input-28-3c8aa3be7db5> in <module>()
----> 1 nth_prime(Integer(10)**Integer(32)+Integer(1))
```

```
    /usr/local/SageMath/local/lib/python2.7/site-packages/sage/arith/misc.pyc in nth_prime
   3611     if n <= 0:
   3612         raise ValueError("nth prime meaningless for non-positive n (=%s)" % n)
-> 3613     return ZZ(pari.prime(n))
   3614
   3615 def quadratic_residues(n):
```

```
        cypari2/auto_instance.pxi in cypari2.pari_instance.Pari_auto.prime (cypari2/pari_instan

        OverflowError: Python int too large to convert to C long
```

In [ ]: