Introducción a la programación con Python (3)

Archivos

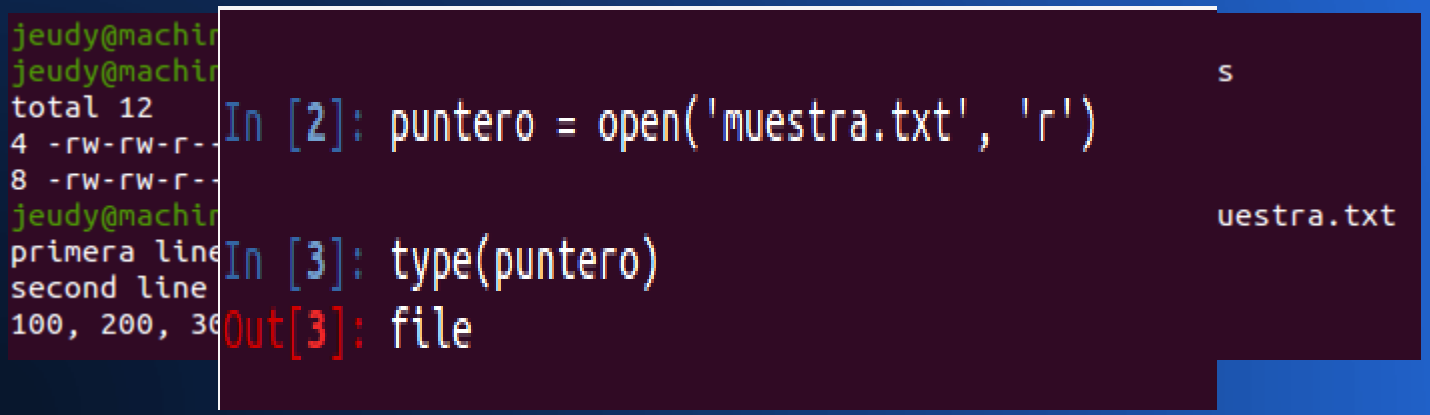
- Python tiene una función **open** para abrir archivos. Devuelve un puntero al archivo.
- Parámetros: path y modo (lectura, escritura, etc)
- S
f

```
In [3]: open?
Type:        builtin_function_or_method
Base Class: <type 'builtin_function_or_method'>
String Form:<built-in function open>
Namespace:  Python builtin
Docstring:
open(name[, mode[, buffering]]) -> file object

Open a file using the file() type, returns a file object.  This is the
preferred way to open a file.  See file.__doc__ for further information.
```

Archivos

- Del repositorio, baje el archivo datos/muestra.txt
- Muevase al directorio datos y abra la consola de python desde ahí.



Modos de apertura de archivos

- 'r': read, solo lectura. El archivo debe existir.
- 'w': write, escritura. Si el archivo no existe, lo crea, si existe, lo sobreescribe (¡OJO!).
- 'a': append. Modo escritura, escribe al final de un archivo existente.
- 'b': para archivos binarios.
- '+' : permite escritura y lecturas simultáneas.

Lectura de archivos

- Asumiendo que abrimos el archivo y creamos una variable puntero:
- puntero.read([n]) : lee todo el archivo a una cadena de texto, o bien los primeros n bytes.
- puntero.readline(): lee una línea a la vez
- puntero.readlines(): lee todas las líneas y devuelve una lista con estas.

Lectura de archivos

```
In [2]: puntero = open('muestra.txt', 'r')

In [8]: puntero.close()

In [9]: type(puntero)
Out[9]: file

In [10]: puntero.read()
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/home/jeudy/Proyectos/UCR/intropython0415/datos/<ipython-input-10-31c9bebd00b7>
in <module>()
----> 1 puntero.read()

ValueError: I/O operation on closed file

Out[6]: 72

In [7]: print contenido
primera linea dentro del archivo
second line in the file
100, 200, 300
```

Lectura de archivos

```
In [12]: puntero = open('muestra.txt', 'r')

In [13]: contenido = puntero.read(10)

In [14]: len(contenido)
Out[14]: 10

In [15]: print contenido
primera li

In [16]: puntero.tell?
Type:        builtin_function_or_method
Base Class: <type 'builtin_function_or_method'>
String Form:<built-in method tell of file object at 0x2e48030>
Namespace:  Interactive
Docstring:  tell() -> current file position, an integer (may be a long integer).

In [17]: puntero.tell()
Out[17]: 10
```

Lectura de archivos

```
In [46]: contenido = puntero.read(20)

In [47]: print contenido
nea dentro del archi

In [48]: contenido = puntero.read(10000)

In [49]: len(contenido)
Out[49]: 42

In [50]: puntero.tell()
Out[50]: 72

In [51]: contenido = puntero.read(10000) #trato de seguir leyendo

In [52]: print len(contenido)
0

In [53]: puntero.tell()
Out[53]: 72
```

Lectura de archivos

```
In [61]: puntero = open('muestra.txt', 'r')

In [62]: lista_lineas = puntero.readlines()

In [63]: type(lista_lineas)
Out[63]: list

In [64]: len(lista_lineas)
Out[64]: 3

In [65]: print lista_lineas
['primera linea dentro del archivo\n', 'second line in the file\n', '100, 200, 3
00\n']
```

Ejercicio

- Cree un programa python archivos_readline.py que reciba un parámetro por línea de comandos (ruta de un archivo de texto).
- Lea las líneas del archivo en una lista.
- Construya una nueva lista, en donde eliminó el caracter \n (nueva línea). Pista: función ***strip*** de los strings.

Lectura de archivos

```
In [78]: puntero = open('muestra.txt', 'r')

In [79]: while True:
   ....:     linea = puntero.readline()
   ....:     print linea
   ....:     if not linea:
   ....:         break
   ....:
primera linea dentro del archivo

second line in the file

100, 200, 300



In [80]: puntero.close()
```

```
In [85]: puntero = open('muestra.txt', 'r')

In [86]: linea = puntero.readline()

In [87]: while linea:
   ....:     print linea
   ....:     linea = puntero.readline()
   ....:
primera linea dentro del archivo

second line in the file

100, 200, 300
```

Escritura de archivos

- La alternativa más sencilla es desde el programa hacer print, que escribe a la salida estándar, y dejar que el usuario redireccione el contenido desde el shell con >
- Baje del repositorio ejemplo_escritura_so.py
- Ejecútelo sin redireccionar
- Ejecútelo redireccionando.

```
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_so.py
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
51,52,53,54,55,56,57,58,59,60
61,62,63,64,65,66,67,68,69,70
71,72,73,74,75,76,77,78,79,80
81,82,83,84,85,86,87,88,89,90
91,92,93,94,95,96,97,98,99,100
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_so.py > datos/numeros.txt
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ cat datos/numeros.txt
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
51,52,53,54,55,56,57,58,59,60
61,62,63,64,65,66,67,68,69,70
71,72,73,74,75,76,77,78,79,80
81,82,83,84,85,86,87,88,89,90
91,92,93,94,95,96,97,98,99,100
```

Escritura de archivos

- puntero = open(ruta, 'w')
- Baje del repositorio ejemplo_escritura_w.py y ejemplo_escritura_a.py
- Note que con 'w', el archivo se sobreescribe cada vez, con 'a', se pegan las líneas al final.

Escritura de archivos

```
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_w.py
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_w.py
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_w.py
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_w.py
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_w.py
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ cat datos/numeros.txt
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
51,52,53,54,55,56,57,58,59,60
61,62,63,64,65,66,67,68,69,70
71,72,73,74,75,76,77,78,79,80
81,82,83,84,85,86,87,88,89,90
91,92,93,94,95,96,97,98,99,100
```

Escritura de archivos

```
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ python ejemplo_escritura_a.py
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$ cat datos/numeros.txt
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
51,52,53,54,55,56,57,58,59,60
61,62,63,64,65,66,67,68,69,70
71,72,73,74,75,76,77,78,79,80
81,82,83,84,85,86,87,88,89,90
91,92,93,94,95,96,97,98,99,100
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
51,52,53,54,55,56,57,58,59,60
61,62,63,64,65,66,67,68,69,70
71,72,73,74,75,76,77,78,79,80
81,82,83,84,85,86,87,88,89,90
91,92,93,94,95,96,97,98,99,100
```

Ejercicio

- Modifique el ejemplo anterior, creando un programa escritura_numeros.py donde se indiquen 3 parámetros:
  - la ruta del archivo a escribir
  - el modo ('w' o 'a')
  - El límite superior del rango de números.

Ejercicio

```
jeudy@machine:~/Proyectos/UCR/intropython0415/ejercicios (master)$ python escritura_numeros.py ../datos/res.dat a 31
jeudy@machine:~/Proyectos/UCR/intropython0415/ejercicios (master)$ python escritura_numeros.py ../datos/res.dat a 51
jeudy@machine:~/Proyectos/UCR/intropython0415/ejercicios (master)$ cat ../datos/res.dat
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
51,52,53,54,55,56,57,58,59,60
61,62,63,64,65,66,67,68,69,70
71,72,73,74,75,76,77,78,79,80
81,82,83,84,85,86,87,88,89,90
91,92,93,94,95,96,97,98,99,100
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
0,1,2,3,4,5,6,7,8,9,10
11,12,13,14,15,16,17,18,19,20
21,22,23,24,25,26,27,28,29,30
31,32,33,34,35,36,37,38,39,40
41,42,43,44,45,46,47,48,49,50
```

Manejo de Errores:

Excepciones

Excepciones

- Errores detectados por el intérprete en tiempo de ejecución.
- Ejemplos:
  - División entre cero
  - Errores de tipos
  - Índice fuera de rango

Ejemplos de excepciones

```
In [101]: print variable_noexiste
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/home/jeudy/Proyectos/UCR/intropython0415/datos/<ipython-input-101-e89177719168> in <module>()
----> 1 print variable_noexiste

NameError: name 'variable_noexiste' is not defined
IndexError: list index out of range
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Manejo de excepciones

- Las excepciones pueden detectarse y manejarlas para que el programa no se interrumpa.
- El programador puede especificar excepciones o bien, decidir dejar pasar todas (no recomendado).
- Para manejar las excepciones en python, usamos bloques try/except

Excepciones comunes

- Exception: tipo general. Atrapa cualquier error.
- OverflowError
- ZeroDivisionError
- AttributeError
- EOFError
- IOError
- ImportError

Excepciones comunes

- IndexError
- KeyError
- NameError
- SyntaxError
- ValueError

Manejo de excepciones

```
In [109]: a = 10

In [110]: b = 0

In [111]: try:
   .....:     c = a / b
   .....: except ZeroDivisionError:
   .....:     print "Detectado error de division por zero. Uso valor por defecto"
   .....:     c = -1
   .....:
Detectado error de division por zero. Uso valor por defecto

In [112]: print c
-1
```

Manejo de Excepciones

- Del repositorio, baje el programa ***manejo_excepciones.py***