Inspira Crea Transforma

UNIVERSIDAD EAFIT®

Programación de Computadores

ST0240

Semestre 2017-1

48 horas semestral

Fundamentos de Programación

ST0286

Semestre 2017-1

48 horas semestral

Agenda

2 clases por Semana

15m Contenido Previo (Tareas, Lecturas, Ejercicios propuestos o Anterior)

30m Contenido de Clase - Entrada y Salida

30m Ejercicios Prácticos

15m Cierre: Trabajo Individual (Por fuera de Clase)

1 clase por Semana

30 m Contenido Previo (Tareas, Lecturas, Ejercicios propuestos o Anterior)

50 m Contenido de Clase - Entrada y Salida

20 m Break

50 m Ejercicios Prácticos

30 m Cierre: Trabajo Individual (Por fuera de Clase)

Entrada, Salida y Ficheros

- Entrada Estándar:
  - Python 2: `raw_input()`
  - Python 3: `input()`
- Salida Estándar: `print`

Entrada y Salida Estándar

- Ejemplos:

```
>>>print “hola \n\n\tmundo”
```

```
>>>for i in range(3):
>>>print i,
0 1 2
```

```
>>>for i in range(3):
>>>print i,
0
1
2
```

```
>>>print “Hola”, “mundo”
Hola mundo

>>>print “Hola” + “mundo”
Holamundo
```

```
>>>print “Cuesta”, 3000, “pesos”
Cuesta 3000 pesos

>>>print “Cuesta” + 3000 + “pesos”
Cuesta 3000 pesos
```

Entrada y Salida estándar

- Ejemplo

```
# coding=utf-8
nombre = raw input("Cómo te llamas? ")
print "Encantado, " + nombre
try:
   edad = raw_input("Cuántos años tienes? ")
   dias = int(edad)*365
   print "Has vivido " + str(dias) + "días"
except ValueError:
   print "Eso no es un número"
```

Entrada y Salida Estándar

- Formatos:

```
>>>print "hola %s" % "mundo"
>>>print "%s %s" % ("Hola","mundo")
```

- Precisión en Números Reales

```
>>>from math import pi
```

- `>>>print "%.4f" % pi`
- `3.1416`

Especificadores de Conversión

Especificador	Formato
%s	Cadena
%d	Entero
%o	Octal
%x	Hexadecimal
%f	Real

Archivos

- Muchos programas de uso diario se denominan “transitorios”, en el sentido de que ellos corren por un breve periodo de tiempo y producen alguna salida, pero cuando para, los datos desaparecen. Si corre el programa nuevamente, éste comienza con “borrón y cuenta nueva”.
- A otros se les denomina “persistentes”: corren por un largo periodo de tiempo (o todo el tiempo), manteniendo al menos algunos datos en almacenamiento permanente (en el disco duro, por ejemplo); y si termina y se reinicia luego, empieza donde quedó en la anterior finalización.
- Una manera simple para mantener los datos es leyendo y escribiendo archivos de texto.
- Otra alternativa es almacenar el estado del programa en una base de datos.

Archivos

```
>>>f = open("<Ruta_Archivo>","<Modo_Acceso>")
```

- Archivos son objetos tipo `file` accesados mediante la función `open`.
- `<Ruta_Archivo>`: Ubicación del archivo con el nombre y la extensión.
- `<Modo Acceso>`: Cadena opcional que indica el modo de acceder al archivo:
- Una vez terminado el trabajo sobre el archivo, se debe cerrar utilizando el método `close`.

Archivos

Modos de Acceso:

- '`r`': read, lectura. Abre el archivo en modo lectura. El archivo tiene que existir previamente, de lo contrario se tiene un error del tipo `IOError`.
- '`w`': write, escritura. Abre el archivo en modo escritura. Si el archivo no existe se crea. Si existe, **SOBREESCRIBIRÁ TODO!!!** el contenido.
- '`a`': append, añadir. Abre el archivo en modo escritura. Continúa escribiendo sobre el archivo existente a partir de la última línea del contenido inicial. No sobreescribe.
- '`b`': binary, binario.
- '`+`': Lectura y Escritura simultáneas.
- '`U`': universal newline, saltos de línea universales. Permite trabajar con archivos que tengan un formato para los saltos de línea que no coinciden con el de la plataforma actual (Windows usa el caracter `CR LF`, Unix, `LF`, y MacOS, `CR`)

Archivos

Lectura de archivos:

- `read`: devuelve una cadena con el contenido del archivo o bien el contenido de los primeros n bytes (si se le especifica).
    - Ej:

```
completo = f.read()
parte = f2.read(512)
```

Archivos

Lectura de archivos:

- `readline`: Sirve para leer las líneas del archivo una por una.
    - Ej:

```
while True:
    linea = f.readline()
    if not linea: break
    print line
```

Archivos

Lectura de archivos:

- `readlines`: Lee todas las líneas del archivo y devuelve una lista con las líneas leídas.

Archivos

Escritura de archivos:

- `write`: Escribe en el archivo una cadena de texto que toma como parámetro.
- `writelines`: Toma como parámetro una lista de cadenas de texto indicando las líneas que queremos escribir en el fichero.

Archivos

Ejemplo:

```
>>>fout = open('output.txt', 'w')  #abre el archivo y lo prepara para escritura
>>>print fout # verifica que efectivamente el archivo fue creado.
<open file 'output.txt', mode 'w' at 0xb7eb2410>
>>>line1 = "Estoy escribiendo la primera línea,\n"
>>>fout.write(line1)  #escribe el contenido de la variable "line1" en el archivo
>>>line2 = "y aquí la segunda.\n"
>>>fout.write(line2)  #escribe el contenido de la variable "line2" en el archivo
>>>fout.close() #Cierra el archivo
```

Taller en Clase

Taller en Clase

Recursos Adicionales

https://spoj.com/

https://ideone.com/

https://www.pythonanywhere.com

http://python.swaroopch.com/

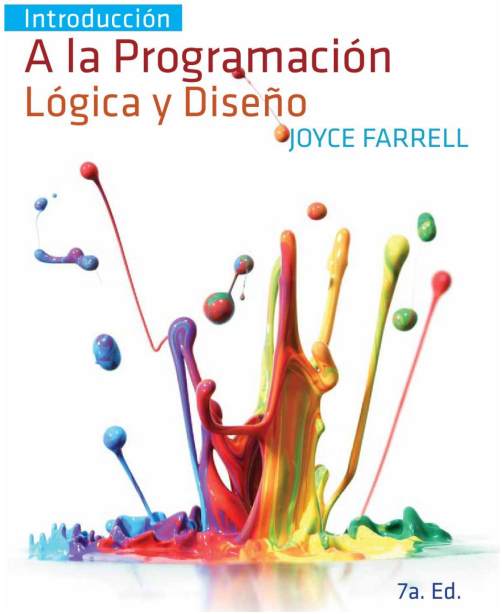
https://pragprog.com/book/gwpy2/practical-programming

https://coderbyte.com/course/learn-python-in-one-week

https://developers.google.com/edu/python/

Para la Próxima Clase/Semana

1. Leer Capítulos 4 y 5 del libro
2. Buscar un algoritmo para generar números primos
3. Consultar que es el triángulo de pascal
4. Exponer problemas que hayan tenido en el foro de la materia.
5. Realizar las tareas dejadas en clase



Referencias y Lecturas Adicionales

- Python Google Class - https://developers.google.com/edu/python/
- Hacker rank - https://www.hackerrank.com/dashboard