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

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

20 m Break

50 m Ejercicios Prácticos

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

Agenda

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

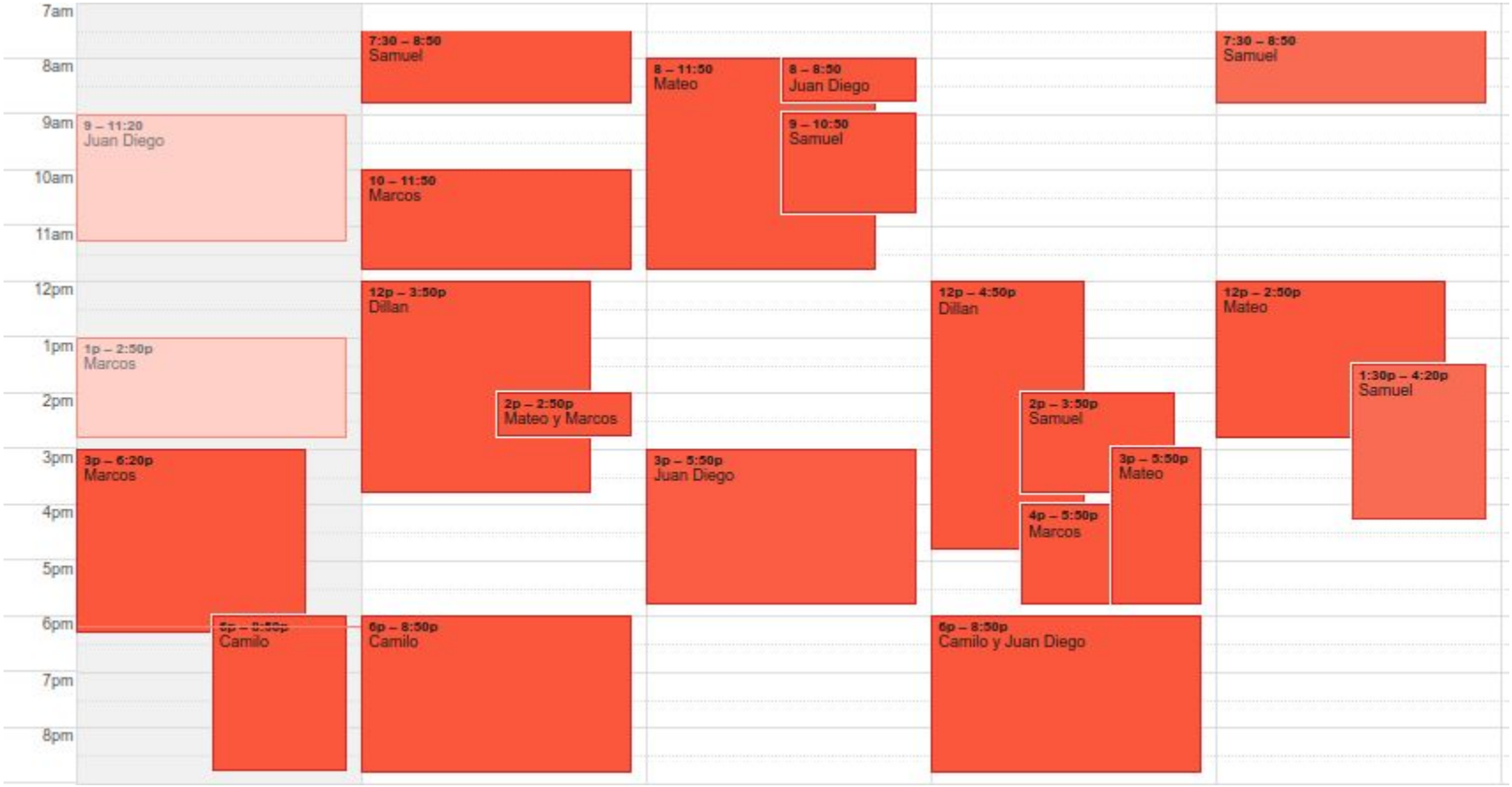
60 m Rangos y Funciones (main)

10 m Break

40 m Ejercicios Prácticos

30 m Taller

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


7am
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
9 – 11:20
Juan Diego
1p – 2:50p
Marcos
3p – 6:20p
Marcos
6p – 8:50p
Camilo
7:30 – 8:50
Samuel
10 – 11:50
Marcos
12p – 3:50p
Dillan
2p – 2:50p
Mateo y Marcos
6p – 8:50p
Camilo
8 – 11:50
Mateo
8 – 8:50
Juan Diego
9 – 10:50
Samuel
3p – 5:50p
Juan Diego
12p – 4:50p
Dillan
2p – 3:50p
Samuel
3p – 5:50p
Mateo
4p – 5:50p
Marcos
6p – 8:50p
Camilo y Juan Diego
7:30 – 8:50
Samuel
12p – 2:50p
Mateo
1:30p – 4:20p
Samuel

Funciones

→ Son fragmentos de código que realizan una secuencia de tareas y posteriormente, entregan un valor. Se identifican por un nombre. E.g., sqrt

→ Los procedimientos hacen referencia a fragmentos de código que poseen un nombre y no devuelven valores. E.g., limpiar pantalla

Funciones

→ En Python no existen los procedimientos dado que si no se especifica un valor de retorno, la función devolverá el valor **None.**

E.g.:

```
def my_function(param1, param2):
    print param1
    print param2
```

¿Cuál es el objetivo de las Funciones?

→ Las funciones cumplen un rol importante a nivel de todos los lenguajes, no solo en Python.

→ Reducen la duplicación de código en un programa.

→ Permiten dividir una tarea o procedimiento de cierta complejidad mediante bloques más pequeños, convirtiéndose cada uno en una función.

→ Mejoran la trazabilidad y la legibilidad del programa.

Lo que se debe tener en cuenta...

→ La función no podrá ejecutarse si no se ha "invocado" o llamado por su nombre.

E.g.:

```
def my_function():
    print "Saludos terrícola"

my_function()   # Llamado de la función
```

Lo que se debe tener en cuenta...

→ La función puede realizar un “retorno” de los datos, los cuales pueden asignarse a una variable.

E.g.:

```
def my_function():
    return "Saludos terrícola"
phrase = my_function()
print phrase
```

Funciones Range y Xrange

→ La función Range se encuentra incorporada en Python, representando así una secuencia de números inmutables.

→ Usualmente esta función se utiliza en un ciclo for para iterar un bloque de código para un número determinado de veces.

Funciones Range y Xrange

⟶ Python v3.x = la función xrange se ha renombrado a la función de range.

⟶ Python v2.x = la función se ha eliminado range en python 3.

Fuente:
http://learntocodewith.me/programming/python/python-2-vs-python-3/

Sintaxis en Range

range([start],stop,[step])

→ El stop es el parámetro requerido.

→ El start especifica dónde debe iniciar el rango.

E.g.: 5 y el valor por defecto es 0.

→ El step especifica el paso entre cada número. El valor por defecto es 1.

Sintaxis en Range

E.g. 1:

```
>>> range(0,10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

E.g. 2:

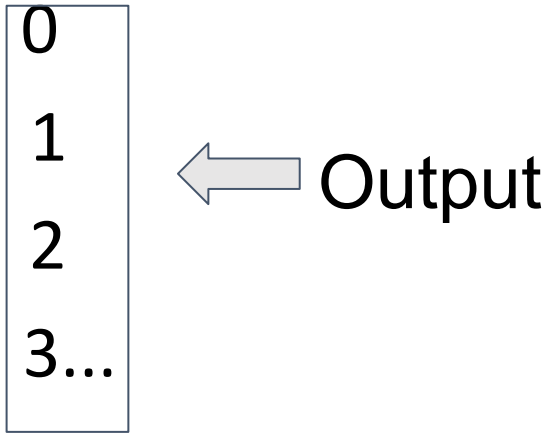
```
>>> code_launching=range(10)
>>> print(code_launching)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Sintaxis en Range

Un for loop para un rango de 100 números. Cada iteración mostrará los números de rango.

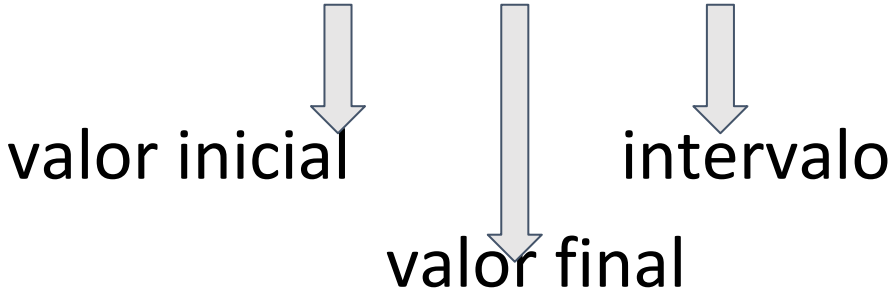
E.g. 3:

```
for k in range(100):
    print(k)
```



Sintaxis en range

range(num1,num2,num3)

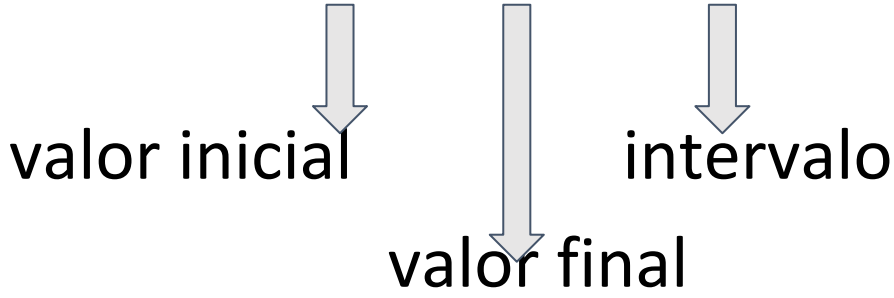


E.g.4: range(10,60,5)

[10, 15, 20, 25, 30, 35, 40, 50, 55]

Sintaxis en xrange

xrange(num1,num2,num3)



E.g.4: xrange(10,60,5)

Son objetos, por lo tanto va a ser más eficiente

Main() Function

→ En algunos lenguajes el hecho de tener sentencias en la parte inferior del programa no es posible. Es por ello que se requiere formen parte de una función especial que es invocada automáticamente por el sistema operativo al momento de ejecutar el programa. Esta función se denomina **main**.

Main() function by Guido V. V. Rossum.

```
"""Module docstring.

This serves as a long usage message.
"""
import sys
import getopt

def main():
    # parse command line options
    try:
        opts, args = getopt.getopt(sys.argv[1:], "h", ["help"])
    except getopt.error, msg:
        print msg
        print "for help use --help"
        sys.exit(2)
    # process options
    for o, a in opts:
        if o in ("-h", "--help"):
            print   doc__
            sys.exit(0)
    # process arguments
    for arg in args:
        process(arg) # process() is defined elsewhere

if   name _ == "__main__":
    main()
```

Fuente:
http://www.artima.com/weblogs/viewpost.jsp?thread=4829

Parámetros

→ Son los valores que espera o recibe una función y que son necesarios para ejecutar las acciones en la misma.

→ Una función puede contener ninguno, uno o más parámetros, los cuales deben ir separados por una coma.

Parámetros

E.g. 1: def my_function(name, last name):

#algorithm

→ Los parámetros se identifican como variables de ámbito local o variables locales donde solo la función podrá tener acceso.

Parámetros

E.g. 2:

```
def my_function(name, last_name):
    complete_name = name, last_name

    print complete_name
```

→ Al llamar una función, sus argumentos deben trasladarse en el mismo orden.

Parámetros por omisión

→ En Python es posible asignar valores por defecto a los parámetros de las funciones. Es decir; la función podrá llamarse con menos argumentos de los esperados.

E.g.:

```
def saludar(nombre, mensaje='Hola'):
    print mensaje, nombre

saludar('Jhon Connor') #imprime: Hola Jhon Connor
```

El uso de las Keywords - Parámetros

→ Otra posibilidad de llamar a una función, se logra trasladando los argumentos esperados, como pares de claves=valor:

```
def saludar(nombre, mensaje='Hola'):
    print mensaje, nombre
saludar(mensaje="Buen día", nombre="Lex Luthor")
```

Parámetros arbitrarios

→ Una función puede recibir un número arbitrario o desconocido de argumentos. Dichos argumentos, llegarán a la función en forma de **tupla.**

→ Los argumentos arbitrarios en una función se definen antecediendo un asterisco(*).

Parámetros arbitrarios

```
E.g.:  def recorrer_param_arbitr(param_fijo, *arbitr):
           print param_fijo
       #Los parámetros arbitrarios se corren como tuplas
           for argum in arbitr
               print argum
       recorrer_param_arbitr('Fixed', 'arbitr 1', 'arbitr 2'…)
```

Parámetros arbitrarios

→ Cuando una función recibirá parámetros fijos y arbitrarios, **los arbitrarios siempre deben suceder a los fijos.**

→ Se pueden obtener parámetros arbitrarios como pares de clave=valor. Para ello, al nombre del parámetro se le deben preceder dos asterisco (**):

```
def recorrer_parametros_arbitrarios(param_fijo, *arbitrarios, **Kwords):
    print argumento_fijo
    for argumento in arbitrarios:
        print argumento


#Los argumentos arbitrarios tipo clave, se recorren como los diccionarios.
```

```
… for clave in kwords:
        print “El valor de”, clave, “es”, kwords[clave]
```

```
recorrer_parametros_arbitrarios(“Fixed”, “arbitrario1”,
“arbitrario2”, “arbitrario3”, clave1=”valor uno”,
clave2=”valor dos”)
```

Desempaquetado de Parámetros

→El desempaquetado llega a ser lo contrario al caso anterior. Es decir; una función espera una lista concreta de parámetros donde no están separados, todo lo contrario, están consolidados en una lista o tupla.

→ Al relacionar el signo (*) se hará referencia al nombre de la lista o tupla, la cual se traslada como parámetro en el momento de llamar la función.

Desempaquetado de Parámetros

E.g.: def calcular(importe, descuento):

```
    return importe - (importe * descuento / 100)
```

```
datos = [1500, 10]
print calcular(*datos)
```

Desempaquetado de Parámetros

→ Otro caso similar puede resultar cuando al trasladar los valores como parámetros a una función, se encuentren disponibles en un diccionario. Lo que ocasionará que estos, se trasladen a la función precedidos de dos asteriscos (**).

Desempaquetado de Parámetros

E.g.:

```
def calcular(imp, desc):
    return imp - (imp * desc / 100)

datos = {“desc”: 10, “imp”: 1500}
print calcular(**datos)
```

Ejercicio en Clase y por fuera de Clase no. 1

- Haga un programa que muestre los números del 1 al 10000 pero usando range y xrange
- Haga una comparativa del tiempo transcurrido usando la biblioteca time

Ejercicio en Clase y por fuera de Clase no. 2

La NASA desea implementar la fase II de su proyecto Curiosity en unos cuantos meses para su descenso en la superficie de Marte. Pero dicha misión requiere de un programador como usted para automatizar el proceso desde su lanzamiento hasta el aterrizaje. Para ello debe usted tener en cuenta lo siguiente:

1. El calentamiento de los cohetes se realiza durante una ignición un ciclo de 5 pasos.
2. Los astronautas deben ingresar una contraseña de acceso correcta, de lo contrario todo el sistema se apagará.
3. El sistema de oxígeno, temperatura, antiradiación y evacuación son independientes, pero dependen del lanzamiento y del aterrizaje.
4. Cada sistema mencionado posee un proceso de estabilización de 5 segundos cada 3 repeticiones.

Ejercicio en Clase y por fuera de Clase no. 2

*¿Qué debe tener en cuenta para resolver el problema?:

→ Analice e interprete el problema y a continuación aplique la lógica de programación...

→ Realice una declaración de variables, constantes, tipología, inicialización…

→ Implemente algunos de los ciclos o bucles que ya conoce…

→ Tenga en cuenta las funciones, rangos y parámetros...

→ Recuerde también declarar las condiciones y estructuras de tipo secuencial, selectivas o decisión simple…

Taller en Clase (5%)

Actividad Individual, Duración 30 minutos al finalizar las 3 horas semanales.

Objetivos de Evaluación: identificar el avance personal según los objetivos propuestos para las dos primeras semanas

Contenido a Evaluar: Expresiones, Variables, Tipos, Entrada y Salida básica, Decisiones, Ciclos

Recursos Adicionales

https://spoj.com/

https://ideone.com/

https://www.pythonanywhere.com

http://python.swaroopch.com/

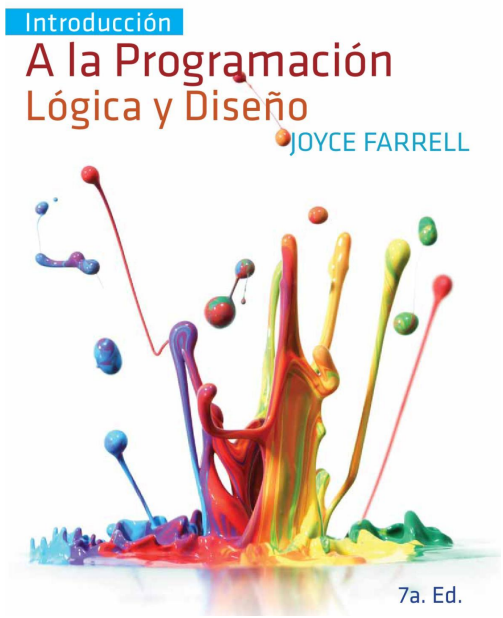
https://pragprog.com/book/gwpy2/practical-programming

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

- History of Computing - http://slideplayer.com/slide/5126850/
- Python Google Cla= - https://developers.google.com/edu/python/
- Computer Programming Khan Academy - https://www.khanacademy.org/computing/computer-programming
- https://code.org
- http://www.skulpt.org/
- https://repl.it/languages/python3
- https://www.khanacademy.org
- https://www.codecademy.com/es/learn/python
- https://www.codeschool.com/courses/try-python/
- https://www.edx.org/course/introduction-computer-science-harvardx-cs50x