Introducción a la computación

1er cuatrimestre de 2016

Python

Operadores booleanos

```
>>> x = 19
>>> if (not x < 3 and True) or x == "Tokio":
...      print "condition holds"
...
condition holds
```

Operador In

```
>>> if (3 in [4, 5, 6, 3]):
...     print "Obvio, no?"
... else:
...     print "WTF?!?!"
...
Obvio, no?
```

Python

Operadores booleanos

```
>>> x = 19
>>> if (not x < 3 and True) or x == "Tokio":
...      print "condition holds"
...
condition holds
```

Operador in

```
>>> if (3 in [4, 5, 6, 3]):
...     print "Obvio, no?"
... else:
...     print "WTF?!?!"
...
Obvio, no?
```

Python

Funciones

```
>>> def fib(n):    # escribe la serie de Fibonacci hasta n
...     a, b = 0, 1
...     while a < n:
...         print a,
...         a, b = b, a+b
...

>>> fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Python

Funciones

```
>>> def fib(n):    # escribe la serie de Fibonacci hasta n
...     a, b = 0, 1
...     while a < n:
...         print a,
...         a, b = b, a+b
...

>>> fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 67, 21]
>>> a[3]
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 67, 21]
>>> a[3]
67
>>> a[2:4]
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 67, 21]
>>> a[3]
67
>>> a[2:4]
[4, 67]
>>> a[0:4:2]
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 67, 21]
>>> a[3]
67
>>> a[2:4]
[4, 67]
>>> a[0:4:2]
[2, 4]
>>> a[::2]
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 67, 21]
>>> a[3]
67
>>> a[2:4]
[4, 67]
>>> a[0:4:2]
[2, 4]
>>> a[::2]
[2, 4, 21]
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 5]
>>> a + [6]
[2, 3, 4, 5, 6]
>>> a
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 5]
>>> a + [6]
[2, 3, 4, 5, 6]
>>> a
[2, 3, 4, 5]
>>> a.append(6)
>>> a
```

Python

Más sobre listas

```
>>> a = [2, 3, 4, 5]
>>> a + [6]
[2, 3, 4, 5, 6]
>>> a
[2, 3, 4, 5]
>>> a.append(6)
>>> a
[2, 3, 4, 5, 6]
```

Python

Más sobre listas

`list.extend(L)`: extiende la lista agregándole todos los ítems de la lista dada;

`list.insert(i, x)`: inserta un ítem en una posición dada (el primer argumento es el índice del ítem delante del cual se insertará);

`list.remove(x)`: quita el primer ítem de la lista cuyo valor sea x (da error si no existe tal ítem);

`list.index(x)`: devuelve el índice en la lista del primer ítem cuyo valor sea x (Da un error si no existe tal ítem);

`list.count(x)`: devuelve el número de veces que x aparece en la lista.

Python

Ejercicio. Escribir una función *prefijos* que dada una lista A, devuelva la lista de todos los prefijos de A.

Ej:

```
prefijos([1, 4, 5]) = [[], [1], [1,4], [1,4,5]]
```

Python

Iteración de listas

```
>>> a = ["hola", 45, [34]]
>>> for x in a:
...     print x
...
hola
45
[34]
```

Python

Rangos

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

>>> range(5, 10)
[5, 6, 7, 8, 9]

>>> range(0, 10, 3)
[0, 3, 6, 9]
```

Python

Rangos

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

>>> range(5, 10)
[5, 6, 7, 8, 9]

>>> range(0, 10, 3)
[0, 3, 6, 9]
```

Python

Rangos

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

>>> range(5, 10)
[5, 6, 7, 8, 9]

>>> range(0, 10, 3)
[0, 3, 6, 9]
```

Python

Strings y listas

```
>>> lista = ['a', 'b', 'c', 'd']

>>> '-'.join(lista)

'a-b-c-d'
```

Python

Strings

```
>>> s = 'hola'

>>> s.replace('h', 'b')

'bola'

>>> s

'Hola'

>>>
```

¿Y si quisiéramos guardar el resultado de hacer?

`s.replace('h','b')`

Python

Diccionarios

- conjunto no ordenado de pares *clave: valor*
- claves: cualquier valor inmutable

```
>>> tel = {"juan": 47813639, "carlos": 47746187}
>>> tel["guido"] = 47855730
>>> tel
{'juan': 47813639, 'carlos': 47746187, 'guido': 47855730}

>>> tel["juan"]
47813639

>>> del tel["guido"]
>>> tel
{'juan': 47813639, 'carlos': 47746187}
```

Python

Diccionarios

- conjunto no ordenado de pares *clave: valor*
- claves: cualquier valor inmutable

```
>>> tel = {"juan": 47813639, "carlos": 47746187}
>>> tel["guido"] = 47855730
>>> tel
{'juan': 47813639, 'carlos': 47746187, 'guido': 47855730}

>>> tel["juan"]
47813639

>>> del tel["guido"]
>>> tel
{'juan': 47813639, 'carlos': 47746187}
```

Python

Diccionarios

- conjunto no ordenado de pares *clave: valor*
- claves: cualquier valor inmutable

```
>>> tel = {"juan": 47813639, "carlos": 47746187}
>>> tel["guido"] = 47855730
>>> tel
{'juan': 47813639, 'carlos': 47746187, 'guido': 47855730}

>>> tel["juan"]
47813639

>>> del tel["guido"]
>>> tel
{'juan': 47813639, 'carlos': 47746187}
```

Python

Diccionarios

- conjunto no ordenado de pares *clave: valor*
- claves: cualquier valor inmutable

```
>>> tel = {"juan": 47813639, "carlos": 47746187}
>>> tel.keys()
['juan', 'carlos']

>>> for x in tel.keys():
...     print x, tel[x]
...
juan 47813639
carlos 47746187
```

Python

Diccionarios

- conjunto no ordenado de pares *clave: valor*
- claves: cualquier valor inmutable

```
>>> tel = {"juan": 47813639, "carlos": 47746187}
>>> tel.keys()
['juan', 'carlos']

>>> for x in tel.keys():
...     print x, tel[x]
...
juan 47813639
carlos 47746187
```

Python

Manejo de archivos

- binarios: .doc, .exe, .jpg
- texto: .txt, .log, .c, .py

```
f = open('pepe.txt', 'r')
```

- open(archivo, modo)
  - modos: 'r', 'w', 'a' (read, write, append)
- procesamiento secuencial

Python

Lectura de archivos: read()

```
>>> f = open('archivo1.txt', 'r')
>>> f.read()
'Este es el archivo entero.\n'
>>> f.read()
''
```

Ojo: el archivo puede ser muy grande

Python

Lectura de archivos: readline()

```
>>> f = open('archivo2.txt', 'r')
>>> f.readline()
'Esta es la primer linea del archivo.\n'
>>> f.readline()
'Segunda linea del archivo\n'
>>> f.readline()
''
```

Lee una línea por vez

Python

Lectura de archivos: readlines()

```
>>> f = open('archivo2.txt', 'r')
>>> f.readlines()
['Esta es la primer linea del archivo.\n', 'Segunda linea del
archivo\n']
```

Python

Lectura de archivos: buena alternativa!

```
>>> f = open('archivo2.txt', 'r')
>>> for linea in f:
...     print linea,

Esta es la primer linea del archivo
Segunda linea del archivo
```

Python

Escritura de archivos: write

```
>>> f = open('out.txt', 'w')
>>> f.write('Esto es una prueba')
>>> f.close()

$ cat out.txt
Esto es una prueba
```

Sólo escribe strings. Para escribir otros valores, primero hay que convertirlos a strings.

```
>>> f = open('out.txt', 'w')
>>> f.write(str(5))
>>> f.close()
```

Python

Ejercicio.

Escribir un programa en Python que dado un archivo de un programa escrito en Python devuelva la cantidad de líneas de código del mismo. Deben ignorarse líneas en blanco y aquellas que sólo contengan comentarios.

Se sugiere usar las funciones de strings str. lstrip y str.startswith

Python