

# REPASO

Programación con Python 2.7+

# Conocimientos básicos

- Tipos de datos
  - Números, cadenas booleanos
    - 5, 3.714, “hola mundo”, True (1), False (0)
  - Función ***type***
    - `type(“hola”), type(5), type(5.0)`
- Colecciones
  - Listas
    - `l = [1,2,3,4]`
  - Tuplas
    - `t = (1,2,3,4)`
  - Diccionarios
    - `D = {“color”: “Rojo”, “Altura”: 170}`

# Control de Flujo

- Sentencias condicionales

- If

```
fav = "mundogeek.net"  
# si (if) fav es igual a "mundogeek.net"  
if fav == "mundogeek.net":  
    print "Tienes buen gusto!"  
    print "Gracias"
```

- If ... else

```
if fav == "mundogeek.net":  
    print "Tienes buen gusto!"  
    print "Gracias"  
else:  
    print "Vaya, que lástima"
```

- If ... elif ... elif ... else

```
if numero < 0:  
    print "Negativo"  
elif numero > 0:  
    print "Positivo"  
else:  
    print "Cero"
```

- A if C else B

```
var = "par" if (num % 2 == 0) else "impar"
```

# Control de flujo

- Bucles

- while

```
edad = 0
while edad < 18:
    edad = edad + 1
    print "Felicidades, tienes " + str(edad)
```

```
while True:
    entrada = raw_input("> ")
    if entrada == "adios":
        break
    else:
        print entrada
```

- for .. in

```
int mi_array[] = {1, 2, 3, 4, 5};
int i;
for(i = 0; i < 5; i++) {
    printf("%d\n", mi_array[i]);
}
```

```
secuencia = ["uno", "dos", "tres"]
for elemento in secuencia:
    print elemento
```

# Funciones

```
def mi_funcion(param1, param2):  
    """Esta funcion imprime los dos valores pasados  
    como parametros"""  
    print param1  
    print param2
```

```
mi_funcion("hola", 2)
```

```
mi_funcion(param2 = 2, param1 = "hola")
```

```
def imprimir(texto, veces = 1):  
    print veces * texto
```

```
>>> imprimir("hola")  
hola
```

```
>>> imprimir("hola", 2)  
holahola
```

```
def varios(param1, param2, *otros):  
    for val in otros:  
        print val
```

```
varios(1, 2)  
varios(1, 2, 3)  
varios(1, 2, 3, 4)
```

# Ejercicio 1

- Escribir un programa que pida dos números enteros e imprima True si la división es exacta, Falso si no.

```
1  # -*- coding: utf-8 -*-
2
3  |
4  def divisible(x, y):
5      return True if x % y == 0 else False
6
7
8  if __name__ == "__main__":
9      x = input("Ingrese un numero entero ")
10     y = input("Ingrese otro numero entero ")
11
12     print(divisible(x, y))
13
```

# Ejercicio 2

- Escriba una función que pida dos números e imprima cual es el mayor, cual es el menor o si son iguales

```
1  # -*- coding: utf-8 -*-
2
3
4  def compara(x, y):
5
6      if x == y:
7          print("son iguales")
8      elif x < y:
9          print("{} es mayor que {}".format(y, x))
10     else:
11         print("{} es mayor que {}".format(x, y))
12
13
14  if __name__ == "__main__":
15     x = input("ingrese primer número: ")
16     y = input("ingrese segundo número: ")
17     compara(x, y)
18
```

# Ejercicio 3

- Escriba una función que reciba una variable entera *size* y retorne en una lista todos los números que están entre 1 y *size* que sean múltiplos de 3 y 7.

```
1  # -*- coding: utf-8 -*-
2
3
4  def numeros_multiplos(size):
5      multiplos = list()
6      for i in range(size):
7          if (i % 3 == 0 and i % 7 == 0):
8              multiplos.append(i)
9      return multiplos
10
11
12 if __name__ == "__main__":
13
14     size = input("Ingrese un número: ")
15     print(numeros_multiplos(size))
16
```



# Ejercicio 4

- Escriba una función que reciba una lista de palabras separadas por coma y las imprima en un string separadas por coma pero ordenadas alfabéticamente.

```
1  # -*- coding: utf-8 -*-
2  # Solution:
3
4
5  def ordena(palabras):
6      items = [x for x in palabras.split(',')]
7      items.sort()
8      return ','.join(items)
9
10
11 if __name__ == "__main__":
12
13     palabras = raw_input("Ingrese una lista de palabras separadas por coma: ")
14     print(ordena(palabras))
15
```

# Ejercicio 5

- Escriba un programa que reciba N líneas (hasta que el usuario ingrese una línea vacía) y las retorne todas en mayúsculas.

```
1  # -*- coding: utf-8 -*-
2  # Solution:
3
4
5  def mayusculas(lineas):
6      lista = list()
7      for frase in lineas:
8          lista.append(frase.upper())
9
10     return lista
11
12
13 if __name__ == "__main__":
14     lines = []
15     while True:
16         s = raw_input("Ingrese una frase y presione ENTER (dejar en blanco para salir): ")
17         if s:
18             lines.append(s)
19         else:
20             break;
21
22     print(mayusculas(lines))
23
24
```

# Ejercicio 6

- Escriba un programa que acepte una secuencia de palabras separadas por espacio. Imprima las removiendo todas las palabras repetidas y ordenas alfabéticamente. (Tip: Utiliza la estructura set de python)
  - Si la entrada es
    - hello world and practice makes perfect and hello world again
  - Entonces la salida debe ser:
    - again and hello makes perfect practice world

```
1 # -*- coding: utf-8 -*-
2 # Solution:
3
4 s = raw_input()
5 words = [word for word in s.split(" ")]
6 print " ".join(sorted(list(set(words))))
7 |
```

# Tarea

Incribirse en [www.codecademy.com](http://www.codecademy.com) y realizar el curso de python.  
De preferencia versión en Inglés.

codecademy

Learn to code interactively, for free.

Sign up

Login



## Python

Learn to program in Python, a powerful language used by sites like YouTube and Dropbox.

START

2.5m+

enrolled students

13 Hours

estimated course time

Beginner

required technical level

# Excepciones

- Son errores detectados por python durante la ejecución de un programa.
  - Intentar dividir un número entre 0
  - Tratar de acceder a un archivo que no existe
- Si la excepción no se captura el flujo de ejecución se interrumpe y se muestra la información asociada.
- En python se utiliza *try .... except* para capturar y tratar las excepciones

```
try:
    f = file("archivo.txt")
except:
    print "El archivo no existe"
```

```
try:
    num = int("3a")
    print no_existe
except NameError:
    print "La variable no existe"
except ValueError:
    print "El valor no es un numero"
```

```
try:
    num = int("3a")
    print no_existe
except (NameError, ValueError):
    print "Ocurrio un error"
```

```
try:
    num = 33
except:
    print "Hubo un error!"
else:
    print "Todo esta bien"
```

```
try:
    z = x / y
except ZeroDivisionError:
    print "Division por cero"
finally:
    print "Limpiando"
```

# Ejercicio 7

- Escriba un programa que permita crear dos listas de palabras y que, a continuación, elimine de la primera lista los nombres de la segunda lista
  - Primero debe pedir la cantidad de elementos de la 1era lista
  - Luego debe pedirle al usuario que ingrese las palabras
  - Lo mismo con la segunda lista
  - Mostrar la primera lista original, luego al segunda lista y finalmente la primera lista sin las palabras de la segunda lista

# Propuesta (7)

```
def get_int_value(prompt):
    answer = True
    while answer:
        try:
            value = int(input(prompt))
            answer = False
            return value
        except:
            print("Por favor ingrese un valor numerico")

def get_list(list_name):
    size = get_int_value(
        "Ingrese la cantidad de elementos de {}: ".format(
            list_name))
    array = list()
    for i in range(size):
        array.append(raw_input("Ingrese una palabra: "))
    return array

def remove_words(list_a, list_b):
    for word in list_b:
        contains_element = True
        while contains_element:
            try:
                list_a.remove(word)
            except ValueError:
                contains_element = False

if __name__ == "__main__":
    a = get_list("1era lista")
    b = get_list("2da lista")
    temp_a = list()
    temp_a[:] = a[:]
    remove_words(a, b)
```

# Importar

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result

if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

```
>>> import fibo
```

```
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
```

```
>>> fib = fibo.fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

```
>>> from fibo import fib, fib2
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

```
>>> from fibo import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

```
$ python fibo.py 50
1 1 2 3 5 8 13 21 34
```

Si se ejecuta el módulo como un Script se ejecuta el bloque if `__main__ == "__main__"`

```
>>> import fibo
>>>
```

Si se importa el módulo no se ejecuta el bloque if `__main__ == "__main__"`



# Ejercicio 8

- Escriba una función que se llame “**encrypt**” en python que reciba como entrada una palabra y a cada caracter (sólo letras) lo modifique por la letra que está a N posiciones de distancia en el abecedario. Se debe poder especificar el número N de posiciones que se desea correr cada carácter (Parámetro de la función).
- Puede omitir la “ñ”
- Extienda la función para que soporte frases
- Tips:
  - Utilizar módulo string de python.
  - String.ascii\_lowercase

# Solución 8

```
1  # -*- coding: utf-8 -*-
2  import string
3
4
5  def encrypt(word, jump=2):
6      abcd = string.ascii_lowercase
7      encrypt_word = ""
8      for char in word:
9          if char != " ":
10             index = (abcd.index(char.lower()) + jump) % len(abcd)
11             encrypt_word += abcd[index]
12          else:
13             encrypt_word += char
14
15     return encrypt_word
16
17
18 if __name__ == "__main__":
19     print(encrypt("frase encriptada", 10))
20
```

# Ejercicio 9

- Cree una función que aplique el encriptado cenit-polar a una frase.
  - Se cambian las letras de la palabra “cenit” por las letras de la palabra “polar”
  - Las letras que no aparecen se mantiene tal cual
- Tips.
  - Utilice la función replace

C	E	N	I	T
P	O	L	A	R

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	B	P	D	O	F	G	H	A	J	K	N	M	L	Ñ	P	C	Q	T	S	R	U	V	W	X	Y	Z

# Tarea

- Utilice la librería Tkinter que viene con python para crear una UI que le permita a un usuario escribir una frase, seleccionar el método de encriptación (cesar, cenit\_polar) y la app responda con la frase encriptada según la elección.
  - <https://docs.python.org/2/library/tkinter.html>

