bitson

Information Technology Solutions on Networking

# Curso

Introductorio

python™

Introductorio

## ejercitación

**El Zen de Python**

- Bello es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Disperso es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son tan especiales como para quebrantar las reglas.
- Aunque lo práctico gana a la pureza.
- Los errores nunca deberían dejarse pasar silenciosamente.
- A menos que hayan sido silenciados explícitamente.
- Frente a la ambigüedad, rechaza la tentación de adivinar.
- Debería haber una -y preferiblemente sólo una- manera obvia de hacerlo.
- Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.15
- Ahora es mejor que nunca.
- Aunque nunca es a menudo mejor que ya mismo.
- Si la implementación es difícil de explicar, es una mala idea.
- Si la implementación es fácil de explicar, puede que sea una buena idea.
- Los espacios de nombres (namespaces) son una gran idea ¡Hagamos más de esas cosas!

**Tim Peters**

La siguiente es una recopilación de los ejercicios propuestos por Facundo Batista en su blog taniquetil.com.ar y de Diego Carballo en su blog Mi diario Python.

# 1. Introducción

1. Hacer un programa que muestre `"Hola mundo"`, y ejecutarlo para ver el mensaje en la pantalla. ¿Cómo lo ejecutó? ¿Qué otras maneras había de ejecutarlo?

2. Abra el Intérprete Interactivo de Python, y realice algunas acciones simples.
   Ingrese la órden `help()`.
   Ingrese la órden `import this`.

3. Suscríbase a la lista de Python Argentina y mande el primer mail con un `"Hola mundo"`.
   Más instrucciones, aquí. Conéctese por IRC al servidor `irc.freenode.org`, y entre al canal `#pyar`. Ahora ya sabe cómo pedir ayuda! Revise la documentación disponible en esta página: Aprendiendo Python.

4. Hacer un programa que muestre `"Esto es un caño"`. Ejecútelo. ¿Funcionó correctamente? ¿Qué ajustes tuvo que hacer para que la `ñ` se vea bien?

# 2. Tipos de Datos

1. Escriba un programa que muestre la siguiente figura:

```
\  |  /
  @ @
   *
 \"""/
```

2. Siendo `a="Hola"`, `b="mundo"`, `c=87` y `d=2.33145`, armar y mostrar las siguientes cadenas:

```
"Hola mundo"  (usando a y b)
"-Hola-mundo-"  (usando a y b)
"El resultado es: 87"  (usando c)
"El resultado es: 87min (5220seg)"  (usando c ambas veces)
"La temperatura es: 2.3"
```

3. Hacer un programa que le pida un número al usuario (usando la función `"input"`) y muestre ese número menos dos, más dos, multiplicado por dos, dividido por dos, dividido por dos de forma entera, y elevado a la potencia de dos.
   **Por ejemplo:** para el número 7, debería mostrar:
   `5 | 9 | 14 | 3 | 3.5 | 49`

4. Pedirle un número al usuario, elevarlo al cuadrado, y mostrar los dígitos al revés y separados por espacio.
   **Por ejemplo:** si el usuario ingresa `17`, la salida tiene que ser `9 8 2`.

5. Pedirle tres números (que pueden ser con decimales) al usuario, base y altura de un rectángulo, y radio de un círculo. Calcular perímetro y área de cada figura.

6. Pedirle un número con muchos decimales al usuario y mostrarlo redondeado a 3 dígitos decimales, y como entero. ¿Qué tipo de redondeo usó? ¿Por qué?

7. Sumar `0.3` diez veces. Por otro lado, multiplicar `0.3` por diez. ¿El resultado es el mismo? ¿Por qué? ¿Qué tipo de dato debe usar?

8. Pedirle una frase al usuario e imprima, en orden, qué consonantes se utilizaron.
   **Por ejemplo:** si se ingresó `"la rana rené"`, mostrar `"lnr"`.

# 3. Control de Flujo

1. Mostrar los números potencia de `2` menores a `10000` de la siguiente manera:

```
0001
0002
0004
0008
...
8192
```

2. Armar un programa que muestre todas las fichas de dominó, sin repetirlas, una por línea.

3. Pedir al usuario un número entero, y mostrar la lista de `1` a ese número, junto con el correspondiente factorial.
   **Por ejemplo:** si el usuario ingresa **4** (prestar atención a lo prolijo del encolumnado):

```
1  1
2  2
3  6
4 24
```

4. Si listamos todos los números naturales menores a `10` que son multiplos de `3` o `5`, tenemos `3`, `5`, `6` y `9`. La suma de estos múltiplos es `23`. Encontrar y mostrar la suma de todos los multiplos de `3` o `5` menores a `1000`.
   **Resultado:** `233168`.

5. Se escriben todos los números enteros positivos en una lista, ordenados y empezando por el `1` (en la posición `0`), luego se tachan todos los que tienen el dígito `9` al menos una vez, y después se tachan todos los múltiplos de

3. Considerando solamente los números no tachados, ¿qué número queda en la posición un millón?
**Resultado:** `2735546`.

6. Encontrar dos números `X` e `Y` enteros, mayores o iguales que `2`, tales que `XY + YX = 94932` (eso es concatenando los dígitos de `X` e `Y`, no multiplicando).
**Resultado:** `X: 57 Y: 375`.

7. Escribir un programa que le pregunte un número al usuario. Si el número es `5`, que muestre `"Suerte!"`; si el número es mayor a `10`, que muestre `"Grande!"`; para los otros casos que muestre `"Sin suerte, :("`.

8. Pedirle un texto al usuario, y mostrar el promedio de largo (hasta dos decimales) de todas las palabras (separando por espacio, tab, enter, etc, no por signos de ortografía) de ese texto.
**Por ejemplo:** para el input `"Hola, hermano, vine a visitarte"`, la salida sería `5.40`.

9. Un número palíndromo (capicúa) se lee igual en los dos sentidos. El palíndromo más grande formado del producto de números de dos dígitos es `9009` (`91 x 99`). Encontrar el palíndromo más grande formado por el producto de dos números de tres dígitos.
**Resultado:** `906609 (993 x 913)`

10. Escribir un programa que le pregunte un número al usuario. Si el número es `5`, que muestre `"Suerte!"`; si el número es mayor a `10`, que le vuelva a pedir el número al usuario; para los otros casos que muestre `"Sin suerte, :("`.

11. Pedirle un texto al usuario, y hacer una estadística sobre cuantas veces aparece cada caracter, mostrando el resultado de mayor a menor respetando la alineación a la derecha de los números.
**Por ejemplo:** para `"nosotros no somos como los orozco, yo los conozco, son ocho los monos"`, debería mostrar:

```
o 23
  12
s  9
n  5
c  5
m  3
l  3
z  2
r  2
,  2
y  1
t  1
h  1
```

12. Pedirle una dirección IP al usuario, en el formato de cuatro decimales separados por punto (validando que cada valor esté entre 0 y 255), y pasarla a una representación hexadecimal.
**Ej:** `20.20.350.20 ->error; 20.30.10.40 ->141E0A28`

13. Pedirle al usuario una contraseña y compararla con una cadena hardcodeada en el código (`"secreta"`). Si es correcta, mostrar `"Bien!"` y salir; si es incorrecta, volverla a pedir, pero sólo tres intentos, a la tercer falla mostrar `"Incorrecto"` y salir. Además, hacer esperar al usuario entre intento e intento, cada vez más.

# 4. Funciones

1. Hacer una función que recibe un número y contesta `"par"` o `"impar"` en función de si el número es par o no.

2. A la misma función que en el punto anterior, agregarle un docstring (con pruebas incluidas, ver el módulo doctest).

3. Hacer una función que recibe dos números y devuelve `"mayor"` (si el primer número es mayor que el segundo), `"menor"`, o `"iguales"`.

4. Hacer una función que recibe un número y contesta `"primo"` o `"no primo"` en función de si el número es primo o no.

5. Al listar los primeros seis números primos (`2, 3, 5, 7, 11 y 13`), podemos ver que el sexto (6°) primo es `13`. ¿Cuál es el 1000° número?
   **Resultado:** `7919`.

6. Hacer un programa que genere un número entero al azar (módulo `random`) entre `0` y `1000`, y le vaya pidiendo al usuario que ingrese números enteros para adivinarlo. Si el usuario ingresa un número menor que el objetivo, muestra `"Es más alto!"`; si el usuario ingresa uno mayor, muestra `"Es más bajo!"`; si el usuario acierta, muestra `"Viva Python!"`, y termina. Si el usuario no acertó en 7 intentos, que muestre `"Alpiste perdiste! Booo"` y termine.

7. Hacer una función que reciba dos palabras y que imprima línea por línea las primeras, segundas, etc. letras de ambas palabras.
   **Por ejemplo:** llamándola con `"Hola"` y `"mundo"`, el resultado sería:

   ```
   H m
   o u
   l n
   a d
     o
   ```

8. Encuentre la cantidad de enteros entre `1` y `107` para los cuales `n` y `n+1` tienen la misma cantidad de divisores positivos enteros. **Por ejemplo:** `14` tiene el `1`, `2`, `7` y `14`, mientras que `15` tiene `1`, `3`, `5` y `15`.
   **Resultado:** `16`.

9. Cuando en un número la diferencia entre cada par de dígitos consecutivos es uno, se lo llama número STEP (como el `123234`, el `9876787654`, etc.). ¿Cuántos números STEP menores a un millón existen?
   **Resultado:** `458`.

10. Pedirle un texto al usuario y mostrar el mismo texto pero sin las vocales.
**Por ejemplo:** para un input de `"Yo estaba allí"`, debería mostrar `"Y stb ll"`.

11. Armar una función que reciba una tupla y devuelva si la tupla está ordenada (de menor a mayor) o no.

12. Escribir dos funciones que permitan convertir de segundos a horas, minutos y segundos, correspondientemente.

13. Implemente una función con el mismo comportamiento que la función builtin `abs`, es decir, que devuelva el valor absoluto del argumento que reciba (y que funcione con números enteros, flotantes, y complejos).

14. Definir una función`max()` que tome como argumento dos números y devuelva el mayor de ellos. *Es cierto que python tiene una función `max()` incorporada, pero hacerla nosotros mismos es un muy buen ejercicio.*

15. Definir una función `max_de_tres()`, que tome tres números como argumentos y devuelva el mayor de ellos.

16. Definir una función que calcule la longitud de una lista o de una cadena dada. *Es cierto que python tiene la función `len()` incorporada, pero escribirla por nosotros mismos resulta una buena práctica.*

17. Escribir una función que tome un carácter y devuelva `True` si es una vocal, de lo contrario devuelve `False`.

18. Escribir las funciones `sum()` y `multip()` que sumen y multipliquen respectivamente todos los números de una lista.
**Por ejemplo:** `sum([1,2,3,4])` debería devolver `10` y `multip([1,2,3,4])` debería devolver `24`.

19. Definir una función `inversa()` que calcule la inversión de una cadena.
**Por ejemplo:** la cadena `"estoy probando"` debería devolver `"odnaborp yotse"`.

20. Definir una función `es_palindromo()` que reconoce palíndromos (es decir, palabras que tienen el mismo aspecto escritas invertidas).
**Por ejemplo:** `es_palindromo("radar")` tendría que devolver `True`.

21. Definir una función `superposicion()` que tome dos listas y devuelva `True` si tienen al menos 1 miembro en común o devuelva `False` de lo contrario. Escribir la función usando el bucle `for` anidado.

22. Definir una función `generar_n_caracteres()` que tome un entero `n` y devuelva el carácter multiplicado por `n`.
**Por ejemplo:** `generar_n_caracteres(5, "x")` debería devolver `"xxxxx"`.

23. Definir un procedimiento `histograma()` que tome una lista de números enteros e imprima un histograma en la pantalla.
**Ejemplo:** `histograma([4, 9, 7])` debería imprimir lo siguiente:
****
*********
*******

24. La función `max()` y la función `max_de_tres()` del ejercicio anterior sólo van a funcionar para 2 o 3 números. Supongamos que tenemos más de 3 números o no sabemos cuantos números son. Escribir una función `max_in_list()` que tome una lista de números y devuelva el más grande.

25. Escribir una función `mas_larga()` que tome una lista de palabras y devuelva la más larga.

26. Escribir una función `filtrar_palabras()` que tome una lista de palabras y un entero `n`, y devuelva las palabras que tengan mas de `n` caracteres.

27. Escribir un programa que le diga al usuario que ingrese una cadena. El programa tiene que evaluar la cadena y decir cuantas letras mayúsculas tiene.

28. Construir un pequeño programa que convierta números binarios en enteros.

29. Escribir un pequeño programa donde:

    - Se ingresa el año en curso.
    - Se ingresa el nombre y el año de nacimiento de tres personas.
    - Se calcula cuántos años cumplirán durante el año en curso.
    - Se imprime en pantalla.

30. Definir una tupla con `10` edades de personas. Imprimir la cantidad de personas con edades superiores a `20`. Puedes variar el ejercicio para que sea el usuario quien ingrese las edades.

31. Definir una lista con un conjunto de nombres, imprimir la cantidad de comienzan con la letra a. También se puede hacer elegir al usuario la letra a buscar. (Un poco mas emocionante)

32. Crear una función `contar_vocales()`, que reciba una palabra y cuente cuantas letras `"a"` tiene, cuantas letras `"e"` tiene y así hasta completar todas las vocales. Se puede hacer que el usuario sea quien elija la palabra.

33. Escriba una función `es_bisiesto()` que determine si un año determinado es un año bisiesto. Un año bisiesto es divisible por 4, pero no por 100. También es divisible por 400.

# 5. Pensando un poco

1. Escribe un programa que te permita jugar a una versión simplificada del juego **Master Mind**. El juego consistirá en adivinar una cadena de números distintos. Al principio, el programa debe pedir la longitud de la cadena (de 2 a 9 cifras). Después el programa debe ir pidiendo que intentes adivinar la cadena de números. En cada intento, el programa informará de cuántos números han sido acertados (el programa considerará que se ha acertado un número si coincide el valor y la posición). Por ejemplo:

```
Dime la longitud de la cadena: 4
Intenta adivinar la cadena: 1234
```

```
Con 1234 has adivinado 1 valores.
Intenta adivinar la cadena: 1243
Con 1243 has adivinado 0 valores.
Intenta adivinar la cadena: 1432
Con 1432 has adivinado 2 valores.
Intenta adivinar la cadena: 2431
Con 2431 has adivinado 4 valores. Felicidades!
```

2. Escribe un programa que pida dos palabras y diga si riman o no. Si coinciden las tres últimas letras tiene que decir que riman. Si coinciden sólo las dos últimas tiene que decir que riman un poco y si no, que no riman.

3. Has un programa que pida al usuario una cantidad de dólares, una tasa de interés y un numero de años. Muestra por pantalla en cuanto se habrá convertido el capital inicial transcurridos esos años si cada año se aplica la tasa de interés introducida. Recordar que un capital `C` dolares a un interés del `x` por cien durante `n` años se convierte en $C \times (1 + x/100)^n$ (años). Probar el programa sabiendo que una cantidad de `10000` dólares al `4.5%` de interés anual se convierte en `24117.14` dólares al cabo de 20 años.

# 6. Extra

1. Escribir un programa que reciba un nombre de archivo al ejecutarse, lo procese e imprima por pantalla cuántas líneas, cuantas palabras y cuántos caracteres contiene el archivo.

   5.02. Escribir un programa que reciba un nombre de archivo al ejecutarse, y grabe (en otro archivo, mismo nombre más ".bz2"), el contenido comprimido con BZIP2.

   5.03. Arme varias funciones que manejen una base de datos en SQLite; una que cree una tabla con los campos "fecha" "descrip", otra que inserte allí un par de registros, y finalmente una que pida y muestre todos los valores.

   5.04. Abrir la URL "http://www.google.com", leer el contenido, y mostrar los headers y el tamaño de la página leida.