

Clase 2. Funciones y Programas



TRABAJADORES
INFORMÁTICOS

organizados en AGC



Ministerio de Trabajo,
Empleo y Seguridad Social
Argentina

Funciones

¿Qué es una función?



Una función es un bloque de código que puede ser reutilizado.

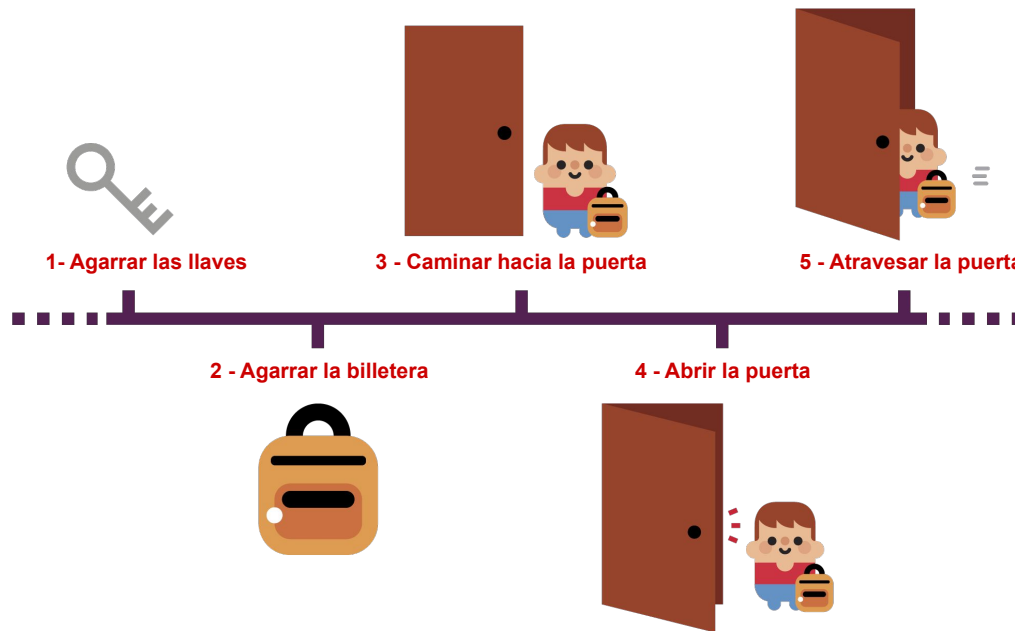


Permiten reutilizar comportamiento, y evitar repetición de código.



Ejercicio

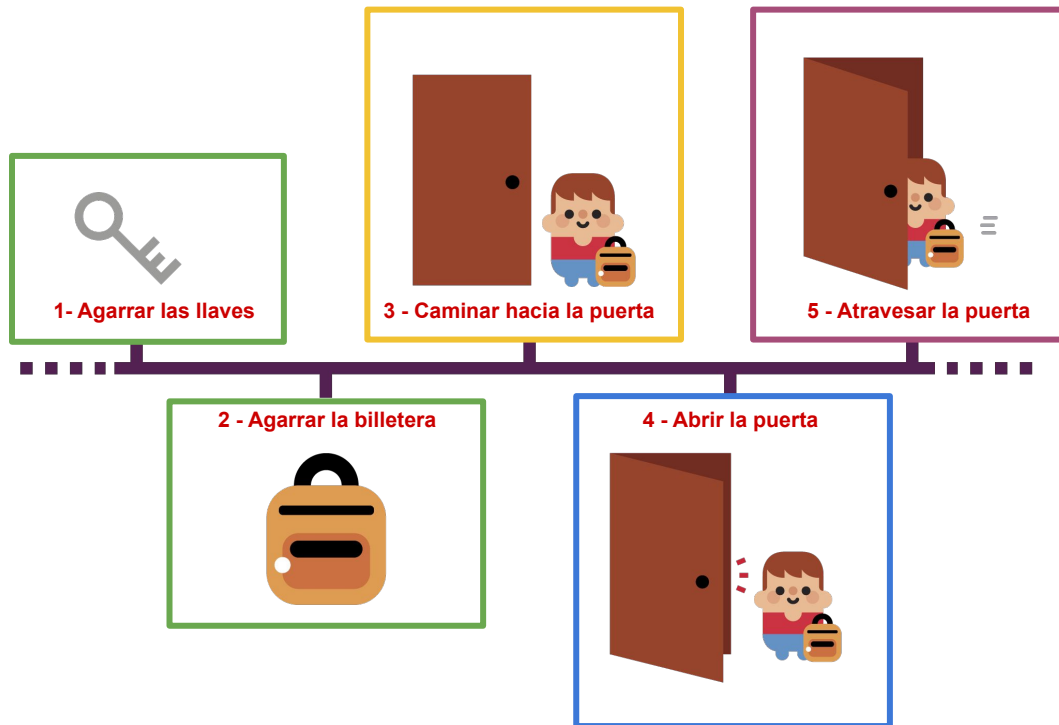
Salir a la calle



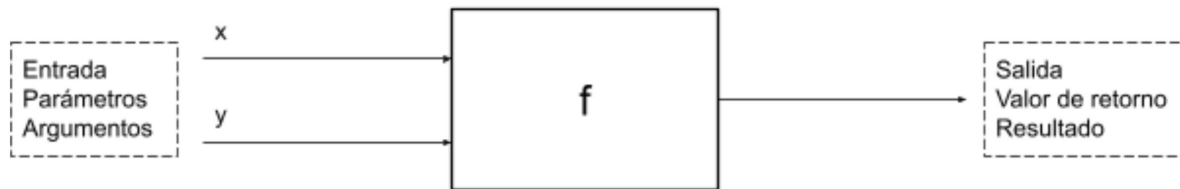
Ejercicio

Salir a la calle

- Agarrar
- Caminar hacia
- Abrir puerta
- Atravesar



¿Qué es una función?

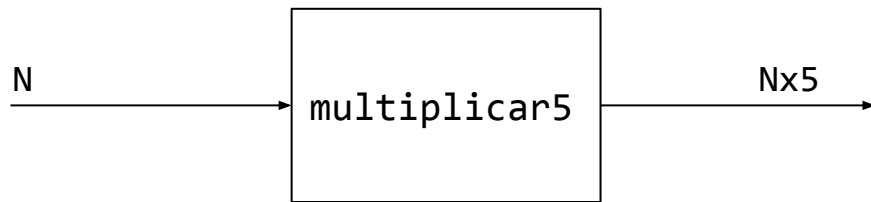


Python 

```
def f(x, y):  
    ...  
    return <resultado>
```

Ejemplo

Multiplicar por 5

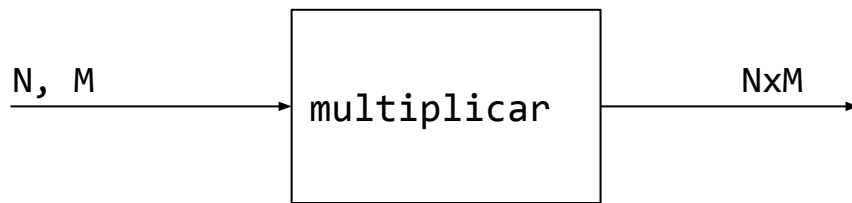


Python 

```
def multiplicar5(n):  
    r = 0  
    for i in range(5):  
        r = r + n  
    return r
```

Ejemplo

Multiplicar por 5



Python 

```
def multiplicar(n, m):  
    r = 0  
    for i in range(m):  
        r = r + n  
    return r
```


Ejercicio

10 mins 



Obtener fizzbuzz

Implementar una función llamada **obtener_fizzbuzz** que reciba un número entero, devuelva Fizz si es múltiplo de 3, Buzz si es múltiplo de 5 (y no es múltiplo de 3), FizzBuzz si es múltiplo de 3 y de 5, o el número recibido en caso contrario

Ejercicio

Obtener fizzbuzz

Python 

```
def obtener_fizzbuzz(n):  
    if n % 3 == 0 and n % 5 == 0:  
        return 'FizzBuzz'  
    elif n % 3 == 0:  
        return 'Fizz'  
    elif n % 5 == 0:  
        return 'Buzz'  
    else:  
        return n
```

Ejercicio

10 mins 



FizzBuzz:

Haciendo uso de la función **obtener_fizzbuzz** implementada en ejercicios anteriores, implementar una función **fizzbuzz** que reciba un número entero n y para todos los números del 0 al n , **imprima** Fizz si es múltiplo de 3, Buzz si es múltiplo de 5 (y no es múltiplo de 3), FizzBuzz si es múltiplo de 3 y de 5, o el número recibido en caso contrario.

Ejercicio

fizzbuzz

Python 

```
def fizzbuzz(n):  
    for i in range(n+1):  
        print(obtener_fizzbuzz(i))
```

Programas

Etapas en la construcción de un programa

Análisis	Entender cuál es el problema que se trata de resolver.
Especificación	Describir detalladamente qué debe hacer el programa (no cómo). En problemas sencillos, puede ser: <ul style="list-style-type: none">- Datos de entrada- Datos de salida- Cómo se relaciona la entrada con la salida
Diseño	Analizar cómo vamos a resolver el problema, cuáles son los algoritmos y las estructuras de datos que vamos a usar. Estudiamos posibles variantes, beneficios y contras de cada una.
Implementación	Traducir a un lenguaje de programación el diseño del paso anterior.
Pruebas	Diseñar un conjunto de pruebas para probar cada parte aislada, y la integración entre ellas.
Mantenimiento	Realizar los cambios necesarios en base a las nuevas demandas, esto puede incluir tanto nuevas funcionalidades como arreglo de bugs.

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Análisis:

¿Qué unidades quiero convertir?

Millas, pies y pulgadas a metros

Especificación:

¿Cuál es la entrada de mi programa?

Millas, Pies y Pulgadas

¿Cuál es la salida de mi programa?

Metros

¿Cómo se relacionan las millas, pies y pulgadas con los metros?

→ $1 \text{ milla} = 1.609344 \text{ km} = 1609.344 \text{ m}$

→ $1 \text{ pie} = 30.48 \text{ cm} = 0.3048 \text{ m}$

→ $1 \text{ pulgada} = 2.54 \text{ cm} = 0.0254 \text{ m}$

→ $M = 1609,344 * L + 0,3048 * F + 0,0254 * P$

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Diseño:

- 01** **Pedirle al usuario** que ingrese millas y guardar respuesta en una variable millas.
- 02** **Pedirle al usuario** que ingrese pies y guardar respuesta en una variable pies

- 03** **Pedirle al usuario** que ingrese pulgadas y guardar respuesta en una variable pulgadas
- 04** Calcular metros ($\text{metros} = 1609.344 * \text{millas} + 0.3048 * \text{pies} + 0.0254 * \text{pulgadas}$) y guardarlo en una variable metros
- 05** **Mostrar** en pantalla metros

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Implementación:


unidades.py 

```
def convertir_a_metros(millas, pies, pulgadas):  
    return 1609.344 * millas + 0.3048 * pies + 0.0254 * pulgadas  
  
def main():  
    print "Convierte medidas inglesas a sistema metrico"  
    L = int(input("Cuántas millas?: "))  
    F = int(input("Y cuántos pies?: "))  
    P = int(input("Y cuántas pulgadas?: "))  
    M = convertir_a_metros(L, F, P)  
    print("La longitud es de ", M, " metros")  
  
main()
```

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Implementación:

unidades.py 


input: La función *input()* permite imprimir por pantalla un mensaje para el usuario, y esperar a que este ingrese un texto.

```
def convertir_a_metros(millas, pies, pulgadas):  
    return 1609.344 * millas + 0.3048 * pies + 0.0254 * pulgadas  
  
def main():  
    print "Convierte medidas inglesas a sistema metrico"  
    L = int(input("Cuántas millas?: "))  
    F = int(input("Y cuántos pies?: "))  
    P = int(input("Y cuántas pulgadas?: "))  
    M = convertir_a_metros(L, F, P)  
    print("La longitud es de ", M, " metros")  
  
main()
```

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Implementación:

unidades.py 

print: La función ***print()*** permite imprimir en pantalla lo que recibe por parámetro.

```
def convertir_a_metros(millas, pies, pulgadas):  
    return 1609.344 * millas + 0.3048 * pies + 0.0254 * pulgadas  
  
def main():  
    print "Convierte medidas inglesas a sistema metrico"  
    L = int(input("Cuántas millas?: "))  
    F = int(input("Y cuántos pies?: "))  
    P = int(input("Y cuántas pulgadas?: "))  
    M = convertir_a_metros(L, F, P)  
    print("La longitud es de ", M, " metros")  
  
main()
```

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Implementación:

unidades.py 

main: Generalmente llamamos **main** a la función principal del programa, es importante que se invoque a esta función para que sea ejecutada.

```
def convertir_a_metros(millas, pies, pulgadas):  
    return 1609.344 * millas + 0.3048 * pies + 0.0254 * pulgadas  
  
def main():  
    print "Convierte medidas inglesas a sistema metrico"  
    L = int(input("Cuántas millas?: "))  
    F = int(input("Y cuántos pies?: "))  
    P = int(input("Y cuántas pulgadas?: "))  
    M = convertir_a_metros(L, F, P)  
    print("La longitud es de ", M, " metros")  
  
main()
```

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Implementación:

unidades.py 

main: Generalmente llamamos **main** a la función principal del programa, es importante que se invoque a esta función para que sea ejecutada.

```
def convertir_a_metros(millas, pies, pulgadas):  
    return 1609.344 * millas + 0.3048 * pies + 0.0254 * pulgadas  
  
def main():  
    print "Convierte medidas inglesas a sistema metrico"  
    L = int(input("Cuántas millas?: "))  
    F = int(input("Y cuántos pies?: "))  
    P = int(input("Y cuántas pulgadas?: "))  
    M = convertir_a_metros(L, F, P)  
    print("La longitud es de ", M, " metros")  
  
if __name__ == '__main__':  
    main()
```

PROGRAMA

ejecutar

Terminal 

```
$ python unidades.py
```

CONSTRUCCIÓN DE UN PROGRAMA

Conversión de unidades

Pruebas:

Probar el programa para valores conocidos, por ejemplo:

- 1 milla, 0 pies, 0 pulgadas
- 0 millas, 1 pie, 0 pulgadas
- 0 millas, 0 pies, 1 pulgada

Mantenimiento:

Detectar bugs:

¿Qué pasa si usamos un float en lugar de un int?

Bug

¿Qué pasa si usamos un string en lugar de un int?

Bug

Ejercicio

10 mins



FizzBuzz:

Haciendo uso de las funciones implementadas en ejercicios anteriores, así como también de las funciones ***input*** y ***print***, implementar un programa que le pida al usuario un número e imprima para todos los números del 0 al ingresado, Fizz si es múltiplo de 3, Buzz si es múltiplo de 5 (y no es múltiplo de 3), FizzBuzz si es múltiplo de 3 y de 5, o el número en caso contrario.

Ejercicio

FizzBuzz

fizzbuzz.py 

```
def fizzbuzz(n):  
    for i in range(n+1):  
        print(obtener_fizzbuzz(i))  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuz(n)  
  
main()
```

Ejercicio

35 mins 



Horas, minutos y segundos:

→ Escribir una función que permita calcular la duración en segundos de un intervalo dado en horas, minutos y segundos.

→ Escribir una función que permita calcular la duración en horas, minutos y segundos de un intervalo dado en segundos.

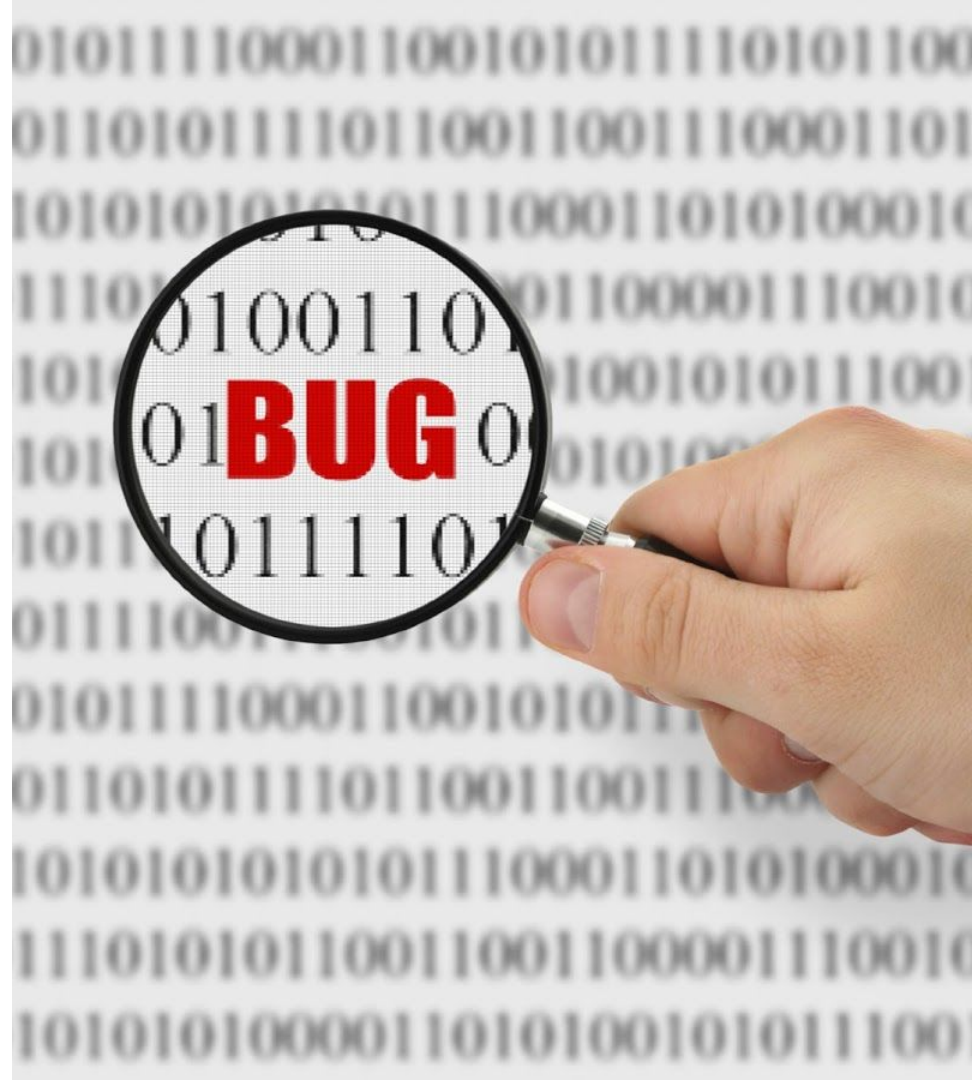
→ Usando las funciones de los puntos a y b, escribir un programa que pida al usuario dos intervalos expresados en horas, minutos y segundos, sume sus duraciones y muestre por pantalla la duración total en horas, minutos y segundos.

Utilizar los conceptos de análisis, especificación y diseño antes de realizar la implementación.

Debugging

¿Qué es un bug?

- ➔ Es un error o falla en un programa, que hace que este tenga resultados incorrectos o inesperados.
- ➔ Llamamos *debugging* al ejercicio de identificar los bugs, para luego solucionarlos.
- ➔ La única forma de aprender a *debuggear* es equivocándose y practicando mucho.



Ejercicio

30 mins



FizzBuzz - Debugging:
Es correcta la siguiente
implementación de Fizzbuzz?

fizzbuzz.py



```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        obtener_fizzbuzz(i)  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

Ejercicio

FizzBuzz - Debugging

Terminal 

```
$ python fizzbuzz.py  
Ingrese un entero: 15
```



Identificamos que nuestro programa
no hace lo que esperamos.

Hay un bug!!!

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        obtener_fizzbuzz(i)  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

Veamos paso a paso qué es lo que hace el programa

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        obtener_fizzbuzz(i)  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

1) Pide entero al usuario

Y si el problema está en el medio? Como podemos verificar si n este tomando el valor correcto?

2) Llama a la función fizzbuzz

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        obtener_fizzbuzz(i)  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    print(n)  
    fizzbuzz(n)  
  
main()
```

1) Pide entero al usuario

2) Llama a la función fizzbuzz

Ejercicio

FizzBuzz - Debugging

Terminal 

```
$ python fizzbuzz.py  
Ingrese un entero: 15  
15
```

Se imprime correctamente **n**,
entonces el problema está
más adelante.

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        obtener_fizzbuzz(i)  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

3) Recorremos del 0 al n, y
llamamos a *obtener_fizzbuzz*

Identificamos que nos falta imprimir
el resultado de *obtener_fizzbuzz* !!!

1) Pide entero al usuario

2) Llama a la función *fizzbuzz*

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        print(obtener_fizzbuzz(i))  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

3) Recorremos del 0 al n, llamamos a *obtener_fizzbuzz* e imprimimos

1) Pide entero al usuario

2) Llama a la función fizz buzz


Ejercicio

FizzBuzz - Debugging

Terminal 

```
$ python fizzbuzz.py
Ingrese un entero: 15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Identificamos que nuestro programa no hace lo que esperamos.

Hay otro bug!!! 

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        print(obtener_fizzbuzz(i))  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

4) Si el número es múltiplo de 3, devolvemos "Fizz"

5) Si el número es múltiplo de 5, devolvemos "Buzz"

6) Si el número es múltiplo de 3 y 5, devolvemos "FizzBuzz"

7) Si no, devolvemos el número

3) Recorremos del 0 al n, llamamos a *obtener_fizzbuzz* e imprimimos

1) Pide entero al usuario

2) Llama a la función fizz buzz

Cómo sabemos si esta entrando o no a cada uno de los "if"?

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        print("%3")  
        return 'Fizz'  
    if n % 5 == '0':  
        print("%5")  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        print("%3 y %5")  
        return 'FizzBuzz'  
    print("No entra a ningun if")  
    return n  
  
...
```

Ejercicio

FizzBuzz - Debugging

Terminal 

```
$ python fizzbuzz.py
Ingrese un entero: 6
No entra a ningun if
0
No entra a ningun if
1
No entra a ningun if
2
No entra a ningun if
3
No entra a ningun if
4
No entra a ningun if
5
```

Sabemos que el problema es
que no está entrando a
ninguno de los **if**

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py



```
def obtener_fizzbuzz(n):  
    if n % 3 == '0':  
        return 'Fizz'  
    if n % 5 == '0':  
        return 'Buzz'  
    if n % 3 == '0' and n % 5 == '0':  
        return 'FizzBuzz'  
    return n  
...
```

4) Si el número es múltiplo de 3, devolvemos "Fizz"

5) Si el número es múltiplo de 5, devolvemos "Buzz"

6) Si el número es múltiplo de 3 y 5, devolvemos "FizzBuzz"

7) Si no, devolvemos el número

Por qué no entra a los **if**?

Identificamos que estamos comparando contra el string '0' en lugar de contra el entero 0

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == 0:  
        return 'Fizz'  
    if n % 5 == 0:  
        return 'Buzz'  
    if n % 3 == 0 and n % 5 == 0:  
        return 'FizzBuzz'  
    return n  
...
```

4) Si el número es múltiplo de 3, devolvemos "Fizz"

5) Si el número es múltiplo de 5, devolvemos "Buzz"

6) Si el número es múltiplo de 3 y 5, devolvemos "FizzBuzz"

7) Si no, devolvemos el número

Ejercicio

FizzBuzz - Debugging

Terminal 

```
$ python fizzbuzz.py
Ingrese un entero: 15
Fizz X
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
Fizz X
```

Nuestro programa tiene mucha mejor pinta, pero todavía hay bugs!

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == 0:  
        return 'Fizz'  
    if n % 5 == 0:  
        return 'Buzz'  
    if n % 3 == 0 and n % 5 == 0:  
        return 'FizzBuzz'  
    return n
```

...

Por qué está devolviendo **Fizz** cuando n es 0 o 15 en lugar de devolver **FizzBuzz**?

Identificamos que importa el orden de los **if**, y que es incorrecto en este caso!

Ejercicio

FizzBuzz - Debugging

fizzbuzz.py 

```
def obtener_fizzbuzz(n):  
    if n % 3 == 0 and n % 5 == 0:  
        return 'FizzBuzz'  
    if n % 3 == 0:  
        return 'Fizz'  
    if n % 5 == 0:  
        return 'Buzz'  
    return n  
  
def fizzbuzz(n):  
    for i in range(n+1):  
        print(obtener_fizzbuzz(i))  
  
def main():  
    n = int(input("Ingrese un entero: "))  
    fizzbuzz(n)  
  
main()
```

4) Si el número es múltiplo de 3 y 5, devolvemos "FizzBuzz"

5) Si el número es múltiplo de 3, devolvemos "Fizz"

6) Si el número es múltiplo de 5, devolvemos "Buzz"

7) Si no, devolvemos el número

3) Recorremos del 0 al n, llamamos a *obtener_fizzbuzz* e imprimimos

1) Pide entero al usuario

2) Llama a la función fizz buzz

Ejercicio

FizzBuzz - Debugging

Terminal 

```
$ python fizzbuzz.py
Ingrese un entero: 15
FizzBuzz
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
```

