Taller 2

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 15-feb-2019 11:59 PM

**Francisco Monsalve **

francisco.monsalve@urosario.edu.co

Instrucciones:

- Guarde una copia de este *Jupyter Notebook* en su computador, idealmente en una carpeta destinada al material del curso.
- Modifique el nombre del archivo del *notebook*, agregando al final un guión inferior y su nombre y apellido, separados estos últimos por otro guión inferior. Por ejemplo, mi *notebook* se llamaría: mcpp_taller2_santiago_matallana
- Marque el *notebook* con su nombre y e-mail en el bloque verde arriba. Reemplace el texto "[Su nombre acá]" con su nombre y apellido. Similar para su e-mail.
- Desarrolle la totalidad del taller sobre este *notebook*, insertando las celdas que sea necesario debajo de cada pregunta. Haga buen uso de las celdas para código y de las celdas tipo *markdown* según el caso.
- Recuerde salvar periódicamente sus avances.
- Cuando termine el taller:
  1. Descárguelo en PDF.
  2. Suba los dos archivos (.pdf y .ipynb) a su repositorio en GitHub antes de la fecha y hora límites.

(El valor de cada ejercicio está en corchetes [ ] después del número de ejercicio.)

---

1. [1]

[Pensar como un computador] Considere el siguiente código:

```
if x > 2: if y > 2: z = x + y print("z es", z) else: print("x es", x)
```

¿Cuál es el resultado si

a) x = 2, y = 5?

b) x = 3, y = 1?

c) x = 1, y = 1?

d) x = 4, y = 3?

El resultado de cada valor es:

a) X es 2.

b) X es 3.

c) X es 1.

d) Z es 7.

In [ ]:

---

2. [1]

[Pensar como un computador] ¿Cuál es el resultado del siguiente código y cuántas veces se recorre el loop?

i = 0 while i < 10: i = i + 1 if i % 2 == 0: print(i)

El resultado del código es: 2, 4, 6, 8, 10.

El código se corre 10 veces, aunque sólo imprime los resultados pares.

In [1]:
```
i = 0
while i < 10:
    i = i + 1
    if i % 2 == 0:
        print(i)
```

```
2
4
6
8
10
```

---

3. [1]

[Pensar como un computador] ¿Cuál es el resultado del siguiente código y cuántas veces se recorre el loop?

i = 0 while i > 10: i = i + 1 if i % 2 == 0: print(i)

El comando no corre/ejecuta debido a que el valor de i=0 no va a cumplir el condicional de while (i=0 siempre será menor a 10), por lo que no tiene resultado ni corre loops

4. [2]

Escriba un programa que pida al usuario ingresar un número entero, y que imprima "par" si el número es par e "impar" si el número es impar. Agregue a su programa un código que genere una advertencia en caso de que el usuario ingrese algo diferente a un número entero: "Error. El usuario debe ingresar un número entero." (Investigue por su cuenta cómo lograr dicha validación y la generación del mensaje.)

In [2]:
```
try:

    num_entero = eval(input())
    if num_entero % 2 == 0:
        print("par")
    elif num_entero % 2 != 0:
        print ("impar")

except NameError:
    print("Error. El usuario debe ingresar un número entero.")
```

```
2
par
```

In [7]:
```
try:

    num_entero = eval(input())
    if num_entero % 2 == 0:
        print("par")
```

```
    elif num_entero % 2 != 0:
        print ("impar")

except NameError:
    print("Error. El usuario debe ingresar un número entero.")
```

```
3
impar
```

In [8]:
```
try:

    num_entero = eval(input())
    if num_entero % 2 == 0:
        print("par")
    elif num_entero % 2 != 0:
        print ("impar")

except NameError:
    print("Error. El usuario debe ingresar un número entero.")
```

```
a
Error. El usuario debe ingresar un número entero.
```

5. [2]

Escriba un for loop que imprima todos los múltiplos de 3 desde 40 hasta 0 en orden decreciente. Esto es, 39, 36, 33,..., 3, 0.

In [3]:
```
for x in reversed (range (41)):
    if (x % 3) == 0:
        print (x)
```

```
39
```

```
36
33
30
27
24
21
18
15
12
9
6
3
0
```

In [ ]:

6. [2]

Escriba un loop que imprima todos los números entre 6 y 30 que no son divisibles por 2, 3 o 5.

In [4]:
```
for y in range (6,31):
    if (y % 2) != 0 and (y % 3) != 0 and (y % 5)!= 0:
        print (y)
```

```
7
11
13
17
19
23
29
```

7. [4]

Escriba un programa llamado "Adivine ni número". El computador generará aleatoriamente un entero entre 1 y 100. El usuario digita un número y el computador responde "Menor" si el número aleatorio es menor que el escogido por el usuario, "Mayor" si el número aleatorio es mayor, y "¡Correcto!" si el usuario adivina el número. El jugador puede continuar ingresando números hasta que adivine correctamente.

Ejemplo:

- El número aleatorio es 79.
- El computador muestra el texto "Adivine el número entre 1 y 100:" y espera a que el usuario lo digite.
- El usuario digita el número que está abajo en itálicas.
- El computador devuelve uno de tres textos, según el caso: "Mayor", "Menor", o "¡Correcto!".

Adivine el número entre 1 y 100: *40*
Mayor
Adivine el número entre 1 y 100: *70* Mayor

Adivine el número entre 1 y 100: *80* Menor

Adivine el número entre 1 y 100: *77* Mayor

Adivine el número entre 1 y 100: *79* ¡Correcto!

¿Cómo generar números aleatorios en Python?

- Al comienzo de su programa escriba: `import random`
- Para generar un número aleatorio entre 1 y 100 escriba: `random.randint(1, 100)`

Pistas:

- Piense en qué estructuras de control le sirven para resolver el problema.

- ¿Cómo determina si el número es mayor, menor o correcto?
- ¿Cómo le da turnos adicionales al usuario para adivinar, dependiendo de si en el turno anterior adivinó o no?

In [5]:
```
import random

num_random = random.randint(1, 100)
guess = int(input("Adivine un número entre 1 y 100: "))

while num_random != guess:

    if guess < num_random:
        print("Mayor")
    elif guess > num_random:
        print ("Menor")
    guess = int(input("Adivine un número entre 1 y 100: "))
    if guess == num_random:
        print("¡Correcto! El número es: ", num_random)
```

```
Adivine un número entre 1 y 100: 50
Menor
Adivine un número entre 1 y 100: 40
Menor
Adivine un número entre 1 y 100: 30
Menor
Adivine un número entre 1 y 100: 20
Mayor
Adivine un número entre 1 y 100: 25
¡Correcto! El número es:  25
```

In [ ]:
```

```