Taller 1

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 7-feb-2020 11:59 PM

Juan Sebastián Gómez

juansebastian.gomezm@urosario.edu.co

Instrucciones:

- Guarde una copia de este *Jupyter Notebook* en su computador, idealmente en una carpeta destinada al material del curso. Sugiero una estructura similar a la del repositorio 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_taller1_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. Esto puede implicar instalar LaTex en su computador. Resuélvalo por su cuenta, por favor. Recuerde: Google es su amigo.
    2. Suba los dos archivos (.pdf y .ipynb) a su repositorio en GitHub antes de la fecha y hora límites. Asegúrese de que Daniel sea "colaborador" de su repositorio y de que los dos archivos queden en su repositorio, en la nube (no solo en su computador). No lo deje para última hora. Talleres subidos después de la fecha y hora límites no serán valorados, como tampoco lo serán si son remitidos vía e-mail.

(Todos los ejercicios tienen el mismo valor.)

---

1. Zelle, sección 1.10 (p. 17):

- "Multiple Choice", Ejercicios # 1-10.
- "Programming Exercises", Ejercicio # 1.

"Multiple Choice" 1-10

1. b
2. d
3. d
4. a
5. b
6. b
7. c
8. b
9. a
10. d

"Programing exercises"

In [19]:

```
#a
print("Hello, world!")
```

```
Hello, world!
```

In [20]:

```
#b
print("Hello", "world!")
```

```
Hello world!
```

In [21]:

```
#c
print(3)
```

```
3
```

In [22]:

```
#d
print(3.0)
```

```
3.0
```

In [23]:

```
#e
print(2 + 3)
```

```
5
```

In [24]:

```
#f
print(2.0 + 3.0)
```

```
5.0
```

In [25]:

```
#g
print("2" + "3")
```

```
23
```

In [26]:

```
#h
print("2 + 3 =", 2 + 3)
```

```
2 + 3 = 5
```

In [27]:

```
#i
print(2 * 3)
```

```
6
```

In [28]:

```
#j
print(2 ** 3)
```

```
8
```

In [29]:

```
#k
print(2 / 3)
```

```
0.6666666666666666
```

En *computer science* son comunes los ejercicios denominados "pensar como un computador". Con estos usted evalúa si está comprendiendo el material, siempre y cuando no utilice un computador para correr el código del enunciado. Siempre que vea un ejercicio marcado con la etiqueta "pensar como un computador", use papel y lápiz o incluso una calculadora si es necesario para descifrar la respuesta, pero nunca ejecute el código en computador.

2. [Pensar como un computador] ¿Cuál es el valor de *w* después de ejecutar el siguiente código?

x = 7 y = 5.0 z = 10.0 w = x % 2 + y / z + z + y / (z + z)

Primero se inicia con la solución del paréntesis

w = x % 2 + y / z + z + y / (z + z)

I. (z + z) = 20.0

w = x % 2 + y / z + z + y / (20)

luego recurrimos a resolver las divisiones (al no haber multiplicaciones) y el módulo en orden de izquierda a derecha

II. x % 2 = 1

w = 1 + y / z + z + y / (20)

III. (y / z) = 0.5

w = 1 + 0.5 + z + y / (20)

IV. (y / (z + z)) = 0.25

w = 1 + 0.5 + z + 0.25

Posteriormente se resulven las sumas en orden de izquierda a derecha

V. 1 + 0.5 = 1.5

w = 1.5 + z + 0.25

VI. 1.5 + z = 11.5

w = 11.5 + 0.25

Se finaliza con

VII. 11.5 + 0.25

w = 11.75

3. [Pensar como un computador] ¿Cuál es el valor de *c* después de ejecutar el siguiente código?

c = True d = False c = c and d c = not c or d

El valor de c es True, porque cuando se indica que c = c(True) y d(False), eso quiere decir que en ese momento c es (False) porque verdadero y falso juntos da como resultado falso. Posteriormente, cuando se indica que c = no c o d, está diciendo que c no es False o False, por lo que c es True.

---

4. Ejecute el siguiente código y responda: ¿Por qué es falsa la tercera línea, mientras que las primeras dos son verdaderas?

1 == 1 "1" == "1" 1 == "1"

In [30]:

```
1 == 1
```

Out[30]:

```
True
```

In [31]:

```
"1" == "1"
```

Out[31]:

```
True
```

In [32]:

```
1 == "1"
```

Out[32]:

```
False
```

Porque el operador relacional "==" está preguntando si el valor de 1 es igual al de "1". Al darse cuenta que un entero (int) 1 y un texto (string) "1" son objetos distintos con valores distintos, se dice que 1 == "1" es falso. Ahora, cuando el operador relacional compara las expresiones en la primera línea, identifica que son iguales y tienen el mismo valor, por lo tanto es verdadero, luego en la segunda línea, repite el mismo procedimiento y ambos objetos también tienen el mismo valor, al ser textos iguales.

5. Escriba un programa que le pida al usuario ingresar su nombre y que arroje un texto saludando de vuelta al usuario, así: "Hola, <nombre>. ¡Veo que aprendes Python rápidamente! ¡Felicitaciones!".

In [11]:

```
nombre = input("Por favor ingrese su nombre: ")
print("¡Hola, " + nombre + ".¡Veo que aprendes Python rápidamente!¡Felicitacione
s!")
```

```
¡Hola, Juan.¡Veo que aprendes Python rápidamente!¡Felicitaciones!
```