Taller 1

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 10-feb-2017 11:59 PM

[David Andres Valles]

[david.valles@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 Juan David 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. What is the fundamental question of computer science?__R// b) what can be computed
2. An algorithm is like a_R// d) recipe
3. A problem is intractable when__R// d) it is not practical to solve
4. Which of the following is not an example of secondary memory?__R// a) RAM
5. Computer languages designed to be used and understood by humans are__R// b) high-level computer languages
6. A statement is_R// b) a complete computer command
7. One difference between a compiler and an interpreter is__R// c) a compiler is not longer needed after a programs is translated
8. By convention, the statements of a program are ofetn placed in a function called___R// b) main
9. Which of the following is not true of comments?__R// a) They make a program more efficient
10. The items listed in the parentheses of a function definition are called___R// d)parameters

Programming Exercises # 1

In [9]:
```
print ("Hello, world!")
```

```
Hello, world!
```

In [10]:
```
print ("Hello", "world!")
```

```
Hello world!
```

In [12]:
```
print(3)
```

```
3
```

In [11]:
```
print(3.0)
```

```
3.0
```

In [13]:
```
print(2+3)
```

```
5
```

In [14]:
```
print(2.0 + 3.0)
```

```
5.0
```

In [15]:
```
print("2" + "3")
```

```
23
```

In [16]:
```
print("2 + 3 =", 2 + 3)
```

```
2 + 3 = 5
```

In [17]:
```
print(2*3)
```

```
6
```

In [19]:
```
print(2**3)
```

```
8
```

In [18]:
```
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)

w = 1 + 0.5 + 10 + 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

Si: c = True, d = False, entonces c= True.

Si: c = True, d = False, c = c and d, entonces c = False (toma la última orden, es decir c = d).

Si: c = True, d = False, c = c and d, c = not c or d, entonces c = True (en el resultado anterior c = False, ahora c no es ni c -False- ni d -False- (c = not c or d),

Por lo tanto c = True

---

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

In [7]:
```
1 == 1
"1" == "1"
```

Out[7]: True

In [8]:
```
1 == 1
"1" == "1"
1 == "1"
```

Out[8]: False

La primera linea se cumple porque es una igualdad entre números, la segunda linea tambien porque es una igualdad entre textos (está entre comillas) y la tercera no se cumple porque no es cierto que el numero 1 sea igual al texto "1".

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 [25]:
```
def ingresar(nombre):
    print("hola", nombre)
    print("¡Veo que aprendes Python rápidamente! ¡Felicitaciones!")
```

In [26]:
```
ingresar ("su nombre acá")
```

```
hola su nombre acá
¡Veo que aprendes Python rápidamente! ¡Felicitaciones!
```