Ejercicios Extra II - Condicionales

Exactas Programa

Verano 2020

Esta serie de actividades están orientadas para practicar los contenidos vistos en la clase de introducción, específicamente la utilización de los condicionales.

> Escribí desde cero cada ejercicio, no copies y pegues el ítem anterior. La idea es que aprendas a escribir en `Python`.
>
> Cada ejercicio que hagas, probalo en el Python Tutor (`http://pythontutor.com`).

Condicionales: if (si), else (si no), elif (si no, si)

La instrucción `if` ejecuta un *camino* de acuerdo a si una condición es verdadera (`True`) y sino (`else`), si era falsa (`False`) ejecuto otro camino.

```
if <condicion>:
    <instrucciones para caso verdadero>
else:
    <instrucciones para caso falso>

<instrucciones sin condicion, fuera del if>
```

Tanto las instrucciones para el caso verdadero como para el falso deberán estar corridas cuatro espacios con respecto al `if` para que sean interpretadas como tales.

En `Python` podría verse algo así (se supone que la variable `mi_variable` está previamente definida con algún valor):

```
1 if mi_variable == 2:
2     mi_variable = mi_variable + 4
3 else:
4     mi_variable = 0
5 otra_variable = 2 * mi_variable
```

En este ejemplo, se ejecutará la instrucción de la línea 2 siempre que `mi_variable` tenga el valor 2, en tanto que se ejecutará la línea 4 en otro caso. Notar que estas dos líneas están *tabuladas*. La línea 5 se ejecutará en cualquiera de los dos casos, tanto si la condición `mi_variable == 2` dio `True` como si dio `False`. ¿Qué valor tendrá `otra_variable` en cada caso?

Ejercicios para hacer

1. Completar y probar:

```
def decir_si_es_mas_grande_que_5(<completar>):
    res = numero > 5
    return res
```

2. Completar y probar:

```
def decir_si_al_dividir_no_da_resto(dividendo, divisor):
    resto = dividendo % divisor
    return <completar>
```

3. Completar y probar:

```
def decir_si_es_negativo(un_numero):
    if <completar - son varias lineas>
    return <completar>
```

4. Completar y probar:

```
def decir_si_la_longitud_es_igual_a(un_nombre, un_numero):
    return <completar>
```

Condiciones combinadas

El `if` *evalúa* la expresión condicional para saber cuál camino tomar. Esta expresión debe *devolver* un valor que sea verdadero o falso (por ejemplo, una condición valida es `a == 4`). Sin embargo, las condiciones no tienen por qué ser comparaciones directas sobre una única variable, se pueden escribir expresiones tan complicadas como uno quiera.

```
1  def sumo_uno_si_es_digito_sino_restar(numero):
2      if numero >= 0 and numero < 10:
3          resultado = numero + 1
4      else:
5          resultado = numero - 1
6      return resultado
7
8  res = sumo_uno_si_digito(4)
9  print(res)
10 res = sumo_uno_si_digito(14)
11 print(res)
```

En este programa, se define una función que utiliza un `if` que sólo suma si se cumplen que el número recibido es, a la vez, mayor o igual a 0 y menor a 10 (es decir, es un dígito decimal) y sino le resta uno al valor recibido. El valor asignado a `res` en la línea 8 será 5, en tanto que el valor asignado a la misma variable en la línea 10 será 13.

Los operadores lógicos que vamos a usar más habitualmente son:

- `and`: devuelve verdadero solo si **las dos** condiciones que lo encierran (que están alrededor) son verdaderas. Por ejemplo, la expresión `2>1 and 2>0` tiene como resultado `True`.
- `or`: devuelve verdadero si **alguna** de las dos condiciones que lo encierran es verdadero. Por ejemplo, la expresión `2<1 or 2>0` dará `True` ya que, a pesar de que la primera condición es falsa, la segunda es verdadera.

Las condiciones se pueden ir combinando y formando expresiones tan complejas como uno quiera. A esta altura del curso, no vamos a profundizar en esto y nos quedaremos en la combinación de un par de expresiones únicamente.

Ejercicios para hacer

En algunos de los siguientes ejercicios, usaremos la función `es_par` que indica si el número recibido es múltiplo de 2:

```
def es_par(un_numero):
    resto = un_numero % 2
    return resto == 0
```

Completar y probar las siguientes funciones:

1.
```
def devolver_valor_mas_grande(valor1, valor2):
    if valor1 > valor2:
        resultado = <completar>
    else:
        resultado = <completar>
    return resultado
```

2.
```
def devolver_el_doble_si_es_par(un_numero):
    if es_par(un_numero):
        <completar>
    else:
        <completar>
    return resultado
```
3.
```
def devolver_segun_condiciones_locas(un_numero):
    if un_numero == 2:
        <completar para que sume 1>
    elif un_numero <= 10:
        <completar para que devuelva el doble>
    elif <completar para que se fije si esta entre 20 y 34>:
        <completar para que sume 5>
    else:
        <completar, que de cero>
    return resultado
```

Más ejercicios para practicar

Definir e implementar las siguientes funciones. En todos los casos se recomienda escribirlas y probarlas con distintos ejemplos para verificar su funcionamiento.

1. Definir una función que recibe un número y devuelve el doble si es más grande que 10 y menor a 20; sino devuelve el mismo
2. Definir una función que recibe un número y devuelve “aprobado!!!” si es más grande que 3 o menor o igual a 10, en caso contrario devuelve “me bocharon”
3. Definir una función que recibe un número y devuelve “Está en el rango deseado” si el valor está entre 5 y 10 y “Fuera de rango” en caso contrario
4. Definir una función que recibe un número y devuelve “Menor a 5” si el valor es menor a 5, “Entre 10 y 20” si el valor está entre 10 y 20, y en cualquier otro caso que devuelva “Número muy grande”