Introducción a la Programación

Guía Práctica 7
Introducción a Lenguaje Imperativo

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Recordar usar las anotaciones de tipado en todas las variables. Por ejemplo: `def funcion(numero: int) ->bool:`.

En los ejercicios se pueden usar funciones matemáticas como por ejemplo: `sqrt, round, floor, ceil, %`. Ver especificaciones de dichas funciones en la documentación de Python: `https://docs.python.org/es/3.10/library/math.html` y `https://docs.python.org/es/3/library/functions.html`

Ejercicio 1. Definir las siguientes funciones y procedimientos:

1. `raizDe2()`: que devuelva la raíz cuadrada de 2 con 4 decimales. Ver función `round`
2. `problema imprimir_hola` () {
   `requiere:` { True }
   `asegura:` { imprime 'hola'por consola}
   }
3. `imprimir_un_verso()`: que imprima un verso de una canción que vos elijas, respetando los saltos de línea.
4. `factorial_de_dos()`

   `problema factorial_2` () : $\mathbb{Z}$ {
   `requiere:` { True }
   `asegura:` {res = 2!}
   }
5. `problema factorial_3` () : $\mathbb{Z}$ {
   `requiere:` { True }
   `asegura:` {res = 3!}
   }
6. `problema factorial_4` () : $\mathbb{Z}$ {
   `requiere:` { True }
   `asegura:` {res = 4!}
   }
7. `problema factorial_5` () : $\mathbb{Z}$ {
   `requiere:` { True }
   `asegura:` {res = 5!}
   }

Ejercicio 2. Definir las siguientes funciones y procedimientos con parámetros:

1. `problema imprimir_saludo` (in nombre: String) {
   `requiere:` { True }
   `asegura:` {imprime "Hola < *nombre* >"por pantalla}
   }
2. `raiz_cuadrada_de(numero)`: que devuelva la raíz cuadrada del número.

3. `imprimir_dos_veces(estribillo)`: que imprima dos veces el estribillo de una canción. Nota: Analizar el comportamiento del operador (*) con strings.

4. `problema es_multiplo_de` (in n: $\mathbb{Z}$, in m:$\mathbb{Z}$) {
   `requiere:` { m $\neq 0$ }
   `asegura:` $\{res = True \leftrightarrow (\exists k : \mathbb{Z})(n = m * k)\}$
   }

5. `es_par(numero)`: que indique si *numero* es par (usar la función `es_multiplo_de()`).

6. `cantidad_de_pizzas(comensales, min_cant_de_porciones)` que devuelva la cantidad de pizzas que necesitamos para que cada comensal coma como mínimo $min_cant_de_porciones$ porciones de pizza. Considere que cada pizza tiene 8 porciones y que se prefiere que sobren porciones.

Ejercicio 3. Resuelva los siguientes ejercicios utilizando los operadores lógicos `and, or, not`. Resolverlos sin utilizar alternativa condicional (if).

1. `alguno_es_0(numero1, numero2)`: dados dos números racionales, decide si alguno de los dos es igual a 0.

2. `ambos_son_0(numero1, numero2)`: dados dos números racionales, decide si ambos son iguales a 0.

3. `problema es_nombre_largo` (in nombre: String) : Bool {
   `requiere:` { True }
   `asegura:` $\{res = True \leftrightarrow 3 \leq |nombre| \leq 8\}$
   }

4. `es_bisiesto(año)`: que indica si un año tiene 366 días. Recordar que un año es bisiesto si es múltiplo de 400, o bien es múltiplo de 4 pero no de 100.

Ejercicio 4. Usando las funciones de python `min` y `max` resolver:
En una plantación de pinos, de cada árbol se conoce la altura expresada en metros. El peso de un pino se puede estimar a partir de la altura de la siguiente manera:

- 3 kg por cada centímetro hasta 3 metros,
- 2 kg por cada centímetro arriba de los 3 metros.

Por ejemplo:

- 2 metros pesan 600 kg, porque 200 * 3 = 600
- 5 metros pesan 1300 kg, porque los primeros 3 metros pesan 900 kg y los siguientes 2 pesan los 400 restantes.

Los pinos se usan para llevarlos a una fábrica de muebles, a la que le sirven árboles de entre 400 y 1000 kilos, un pino fuera de este rango no le sirve a la fábrica.
Definir las siguientes funciones, deducir qué parámetros tendrán a partir del enunciado. Se pueden usar funciones auxiliares si fuese necesario para aumentar la legibilidad.

1. Definir la función `peso_pino`

2. Definir la función `es_peso_util`, recibe un peso en kg y responde si un pino de ese peso le sirve a la fábrica.

3. Definir la función `sirve_pino`, recibe la altura de un pino y responde si un pino de ese peso le sirve a la fábrica.

4. Definir `sirve_pino` usando composición de funciones.

Ejercicio 5. Implementar los siguientes problemas de alternativa condicional (if). Intentá especificarlos alguno de ellos (todos los que te salgan) en lenguaje semiformal y formal sin utilizar IfThenElseFi.

1. `devolver_el_doble_si_es_par(un_numero)`. Debe devolver el mismo número en caso de no ser par.

2. `devolver_valor_si_es_par_sino_el_que_sigue(un_numero)`. Analizar distintas formas de implementación (usando un if-then-else, y 2 if), ¿todas funcionan?

3. `devolver_el_doble_si_es_multiplo3_el_triple_si_es_multiplo9(un_numero)`. En otro caso devolver el número original. Analizar distintas formas de implementación (usando un if-then-else, y 2 if, usando alguna opción de operación lógica), ¿todas funcionan?.
4. Dado un nombre, si la longitud es igual o mayor a 5 devolver una frase que diga "Tu nombre tiene muchas letras!" y sino, "Tu nombre tiene menos de 5 caracteres".
5. En Argentina una persona del sexo femenino se jubila a los 60 años, mientras que aquellas del sexo masculino se jubilan a los 65 años. Quienes son menores de 18 años se deben ir de vacaciones junto al grupo que se jubila. Al resto de las personas se les ordena ir a trabajar. Implemente una función que, dados los parámetros de sexo (F o M) y edad, imprima la frase que corresponda según el caso: "Andá de vacaciones" o "Te toca trabajar".

Ejercicio 6. Implementar las siguientes funciones usando repetición condicional `while`:

1. Escribir una función que imprima los números del 1 al 10.
2. Escribir una función que imprima los números pares entre el 10 y el 40.
3. Escribir una función que imprima la palabra "eco" 10 veces.
4. Escribir una función de cuenta regresiva para lanzar un cohete. Dicha función irá imprimiendo desde el número que me pasan por parámetro (que será positivo) hasta el 1, y por último "Despegue".
5. Hacer una función que monitoree un viaje en el tiempo. Dicha función recibe dos parámetros, "el año de partida" y "algún año de llegada", siendo este último parámetro siempre más chico que el primero. El viaje se realizará de a saltos de un año y la función debe mostrar el texto: "Viajó un año al pasado, estamos en el año: <año>" cada vez que se realice un salto de año.
6. Implementar de nuevo la función de monitoreo de viaje en el tiempo, pero desde el año de partida hasta *lo más cercano* al 384 a.C., donde conoceremos a Aristóteles. Y para que sea más rápido el viaje, ¡vamos a viajar de a 20 años en cada salto!

Ejercicio 7. Implementar las funciones del ejercicio 6 utilizando `for num in range(i,f,p):`. Recordar que la función `range` para generar una secuencia de números en un rango dado, con un valor inicial i, un valor final f y un paso p. Ver documentación: `https://docs.python.org/es/3/library/stdtypes.html#typesseq-range`

Ejercicio 8. Realizar la ejecución simbólica de los siguientes códigos:

1. `x=5 ; y=7`
2. `x=5 ; y=7 ; z=x+y`
3. `x=5 ; x=''hora''`
4. `x=True ; y=False ; res=x and y`
5. `x=False ; res=not(x)`

Ejercicio 9. Sea el siguiente código:

```
def rt(x: int, g: int) -> int:
  g = g + 1
  return x + g

g: int = 0
def ro(x: int) -> int:
  global g
  g = g + 1
  return x + g
```

1. ¿Cuál es el resultado de evaluar tres veces seguidas `ro(1)`?
2. ¿Cuál es el resultado de evaluar tres veces seguidas `rt(1, 0)`?
3. En cada función, realizar la ejecución simbólica.
4. Dar la especificación en lenguaje natural para cada función, rt y ro.