C++ REVIEW DUNE

C++ Review DUNE

Taller de Python para computación científica

Hoja de ejercicios 1

Revisión de Python: estructuras de control, repetición, funciones integradas, funciones definidas por el usuario, NumPy

Ejercicio 1.1 ¿Qué es un *programa*?

Ejercicio 1.2 Python es un *lenguaje interpretado*. ¿Qué significa "interpretado" en este contexto?

Ejercicio 1.3 Asignación:

```
my_int = 5
my_int = my_int + 3
print(my_int)
```

(a) Si ejecuta las tres líneas de código, ¿qué se imprimirá? Explica tu respuesta.

(b) Reescriba `my_int = my_int + 3` usando el símbolo `+=`.

Ejercicio 1.4 Dada la expresión `30 - 3**2 + 8 // 3**2 * 2 * 10`.

(a) ¿Cuál es el resultado de la expresión?

(b) Según la precedencia y la asociatividad de los operadores en Python, coloque correctamente entre paréntesis la expresión de modo que obtenga el mismo resultado que el anterior.

Ejercicio 1.5 Escriba un programa en Python que acepte un número n y calcule el valor de $n+n\cdot n+n\cdot n\cdot n+n\cdot n\cdot n\cdot n$.

Ejercicio 1.6 Considere un *triángulo* con lados de longitud 3u, 7u y 9u. La *ley de cosenos* establece que dados los tres lados de un triángulo a, b y c, el ángulo C entre los lados a y b es $c^2 = a^2 + b^2 - 2ab\cos(C)$. Escriba un programa en Python para calcular los tres ángulos en el triángulo.

Ejercicio 1.7 El radio y la masa de la Tierra son $r = 6378 \times 10^3$ m y $m_1 = 5.9742 \times 10^{24}$ kg, respectivamente. Pedro tiene una masa de X kg. Pida al usuario que ingrese X y luego calcule el módulo de la fuerza gravitacional (F) y el módulo de la aceleración debida a la gravedad (g) causada por la fuerza gravitacional ejercida sobre él por la Tierra. Recuerde, $F = G\frac{m_1 m_2}{r^2}$ y $F = mg$. Sea la constante de gravitación universal $G = 6.67300 \times 10^{-11}$ (en unidades de $\mathrm{m^3 kg^{-1} s^{-2}}$). Comprueba que el valor resultante de g es cercano a 9.8 $\mathrm{ms^{-2}}$.

Ejercicio 1.8 (Usando módulos) Python viene con *cientos de módulos*. Aquí hay un desafío para ti: encuentre un módulo que pueda importar que genere la fecha de hoy para que pueda imprimirlo. Use su motor de búsqueda favorito para obtener ayuda para encontrar qué módulo necesita y cómo usarlo. Al final, su tarea es hacer lo siguiente:

```
>>> print("La fecha de hoy es:", today)
La fecha de hoy es: 2022-11-13
```

Ejercicio 1.9 ¿Cuántos números naturales menores que N son divisibles por 2 y no por 3? Escriba un programa para imprimirlos. *Sugerencia*:

```
the_max = int(input("Ingrese la cota superior:"))
the_sum = 0
extra = 0

for number in range(1, the_max):
    if number % 2 and not number % 3:
        pass
    else:
        pass

print(the_sum)
print(extra)
```

Ejercicio 1.10 ¿Qué es un *iterador*? Dé dos ejemplos de iteradores.

Ejercicio 1.11 Prediga el resultado dado `x = 0`, `y = 2`, `z = 4`.

(a) `x or y`

(b) `x and y`

(c) `x or z`

(d) `(x and y) or (y and z)`

Ejercicio 1.12 Prediga el resultado de cada línea:

```
i = 1
j = 0
while i < 10:
    i += 1
    j += 2 * 2

print(i)
print(j)
```

Ejercicio 1.13 Escriba un programa que imprima todos los años bisiestos desde 2000 hasta 2099 (inclusive). *Sugerencia*:

```
for year in range(2000, 2100):
    if (year % 400 == 0) and (year % 100 == 0):
        pass

    elif (year % 4 == 0) and (year % 100 != 0):
        pass
    else:
        pass
```

Ejercicio 1.14 Complete la siguiente tabla con valores `True` o `False`: un valor en cada cuadro vacío.

p	q	(not p) or q	(p and q) or q	(p or q) and p	(p or q) and (p and q)
True	True				
True	False				
False	True				
False	False				

Ejercicio 1.15 La *fórmula cuadrática* que calcula las raíces para una ecuación cuadrática $ax^2 + bx + c = 0$ es $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Debido a que la raíz cuadrática de un negativo es imaginaria, se puede usar la expresión de la raíz cuadrada (conocida como el *discriminante*) para verificar el tipo de raíz. Si el discriminante es negativo, las raíces son imaginarias. Si el discriminante es cero, solo hay una raíz. Si el discriminante es positivo, hay dos raíces.

(a) Escriba un programa que use la fórmula cuadrática para generar las raíces reales, es decir, ignore las raíces imaginarias. Usa el discriminante para determinar si hay una o dos raíces y luego imprime la respuesta apropiada.

(b) Python usa la letra `j` para representar el número imaginario i (una convención utilizada en ingeniería eléctrica). Sin embargo, la `j` de Python siempre debe ir precedida de un número. Es decir, `1j` es equivalente a la i matemática. Implemente el manejo de raíces imaginarias a su programa.

Ejercicio 1.16 Para entender lo que un programa intenta lograr, es esencial poder seguir el flujo de control. En el siguiente ejemplo, ¿qué sucede cuando `x = 4`? *Sugerencia*: visualize el programa en https://pythontutor.com.

```
while True:
    for x in range(6):
        y = 2 * x + 1
        print(y)
        if y > 9:
            break
```

(a) El programa sale del bucle `while` y deja de ejecutarse.

(b) El programa sale del ciclo `for`, pero la condición `while` sigue siendo verdadera, lo que da como resultado un bucle infinito.

(c) El programa no se interrumpe, simplemente continúa procesando el bucle `for`.

Ejercicio 1.17 Escribe un algoritmo para freír un huevo. Pruébalo con un amigo(a): en una cocina, lee el algoritmo y haz que el amigo(a) haga exactamente lo que dices. ¿Cómo lo hizo?

Ejercicio 1.18 Dada la cadena `m = "Monty Python"`:

(a) Escribe una expresión para imprimir el primer carácter.

(b) Escriba una expresión para imprimir el último carácter.

(c) Escriba una expresión que imprima `Python`.

Ejercicio 1.19 Dada `s = "Earth"`, ¿qué devuelve `s.strip()`, `s.lstrip()` y `s.rstrip()`?

Ejercicio 1.20 ¿Qué hace esta función? ¿Qué devuelve para `number = 5`?

```
def Func(number):
    total = 0
    while number > 0:
        total = total + number * (number - 1)
        number = number - 1
    return total
```

Ejercicio 1.21 ¿Qué hace esta función? ¿Qué devuelve si `x = 5`?

```
def Func(x):
    total = 0
    for i in range(x):
        total += i * (i - 1)
    return total
```

Ejercicio 1.22 ¿Qué hace esta función? ¿Qué número devuelve esta función?

```
def Func():
    number = 1.0
    total = 0

    while number < 100:
        total = 1 // number
        number += 1
    return total
```

Ejercicio 1.23 Escribe una función que tome la masa como entrada y devuelva su energía equivalente. $(E = mc^2.)$

Ejercicio 1.24 La *sucesión de Fibonacci* es 1, 1, 2, 3, 5, 8, 13, ... Puedes ver que el primer y el segundo número son ambos 1. A partir de entonces, cada número es la suma de los dos números anteriores.

(a) Escribe una función para imprimir los primeros N números de la sucesión de Fibonacci.

(b) Escribe una función para imprimir el número N de la secuencia.

Sugerencia:
https://docs.python.org/es/3/tutorial/introduction.html#first-steps-towards-programming

Ejercicio 1.25 Has decidido programar una calculadora sencilla. Su calculadora puede realizar las siguientes operaciones:

(a) Puede sumar dos números.

(b) Puede restar un número de otro.

(c) Puede multiplicar dos números.

(d) Puede dividir un número por otro.

Escriba una función que le pida al usuario dos números y la operación que desea realizar en estos dos enteros, y devuelva el resultado. *Nota*: Las opciones deben ingresarse como cadenas.

Ejercicio 1.26 Escriba una función que acepte una lista de cadenas y las ordene alfabéticamente.
Sugerencia:

```
my_list = ["apple", "box", "tickle", "python", "button"]
# use el comando help(list) y reemplace foo por un método apropiado
my_list.foo()
print(my_list)
```

Ejercicio 1.27 ¿Qué es NumPy?

Ejercicio 1.28 ¿Qué es un arreglo y en qué se diferencia de una lista? ¿Cuál es el nombre de la clase de arreglo integrada en NumPy?

Ejercicio 1.29 Cree los siguientes arreglos NumPy:

(a) Un arreglo 1-D llamado `ceros` teniendo 10 elementos y todos los elementos son igual a cero.

(b) Un arreglo 1-D llamado `vocales` teniendo los elementos `"a"`, `"e"`, `"i"`, `"o"` y `"u"`.

(c) Un arreglo 2-D llamado `unos` teniendo filas y 5 columnas y todo los elementos son igual a uno y `dtype = int`.

(d) Use listas anidadas de Python para crear un arreglo 2-D llamado `myarray1` que tenga 3 filas y 3 columnas y almacene los siguientes datos

$$\begin{bmatrix} 3.7 & -1 & -14 \\ 0 & 2.24 & 1.9 \\ -10.1 & 1 & 3 \end{bmatrix}.$$

(e) Un arreglo 2-D llamado `myarray2` usando `np.arange()` que tiene 3 filas y 5 columnas con valor `start = 4`, `step = 4` y `dtype=float`.

Ejercicio 1.30 Usando los arreglos creados en el **Ejercicio 1.29**, escribe comandos NumPy para lo siguiente:

(a) Encuentre las *dimensiones*, la *forma*, el *tamaño*, el tipo de dato de los elementos y el tamaño de los elementos de los arreglos `ceros`, `vocales`, `unos`, `myarray1` y `myarray2`.

(b) Reagrupe el arreglo de `unos` para tener los 10 elementos en una sola fila.

(c) Muestre el segundo y tercer elemento del arreglo `vocales`.

(d) Muestre los elementos de la segunda y tercera fila del arreglo `myarray1`.

(e) Muestre los elementos de la primera y segunda columna del arreglo `myarray1`.

(f) Muestre los elementos de la primera columna de la segunda y tercera fila del arreglo `myarray1`.

(g) Invierta el arreglo `vocales`.

Ejercicio 1.31 (Operaciones aritméticas) Usando los arreglos creados en el **Ejercicio 1.29**, escribe comandos NumPy para lo siguiente:

(a) Divide todos los elementos del arreglo `unos` por 3.

(b) Sume los arreglos `myarray1` y `myarray2`.

(c) Reste `myarray1` de `myarray2` y almacene el resultado en un nuevo arreglo.

(d) Multiplique `myarray1` y `myarray2` elemento por elemento.

(e) Realice la multiplicación de arreglos `myarray1` y `myarray2` y almacene el resultado en un nuev arreglo `myarray3`.

(f) Divide `myarray1` por `myarray2`.

(g) Encuentre el cubo de todos los elementos de `myarray1` y divida el arreglo resultante por 2.

(h) Encuentre la raíz cuadrada de todos los elementos de `myarray2` y divida el arreglo resultante por 2. El resultado debe redondearse a dos decimales.

Ejercicio 1.32 Usando los arreglos creados en el **Ejercicio 1.29**, escribe comandos NumPy para lo siguiente:

(a) Encuentra el arreglo transpuesto de `unos` y `myarray2`.

(b) Ordena el arreglo `vocales` al revés.

(c) Ordene el arreglo `myarray1` de modo que traiga el menor valor de la columna en la primera fila y así sucesivamente.

Ejercicio 1.33 Usando los arreglos creados en el **Ejercicio 1.29**, escribe comandos NumPy para lo siguiente:

(a) Use `np.split()` para dividir el arreglo `myarray2` en 5 arreglos en forma de columna. Almacene sus arreglos resultantes en `myarray2A`, `myarray2B`, `myarray2C`, `myarray2D` y `myarray2E`. Imprima los arreglos `myarray2A`, `myarray2B`, `myarray2C`, `myarray2D` y `myarray2E`.

(b) Divida el arreglo `ceros` de la matriz en el índice del arreglo 2, 5, 7, 8 y almacene los arreglos resultantes en `cerosA`, `cerosB`, `cerosC` y `cerosD` e imprímalas.

(c) Concatene los arreglos `myarray2A`, `myarray2B` y `myarray2C` en un arreglo que tenga 3 filas y 3 columnas.

Ejercicio 1.34 Cree un arreglo 2-D llamado `myarray4` usando `np.arange()` que tenga 14 filas y 3 columnas con `start = -1`, `step = 0.25`. Divida este arreglo por filas en 3 partes iguales e imprima el resultado.

Ejercicio 1.35 Usando `myarray4` creado en la pregunta de arriba, escribe comandos para lo siguiente:

(a) Encuentra la suma de todos los elementos.

(b) Encuentra la suma de todos los elementos por filas

(c) Encuentra la suma de todos los elementos por columnas.

(d) Encuentra el máximo de todos los elementos.

(e) Encuentre el mínimo de todos los elementos en cada fila.

(f) Encuentra la media de todos los elementos en cada fila.

(g) Encuentre la desviación estándar por columnas.