Unit 1

Python Essentials: a procedural approach

1.1 Exercises

Exercise 1.1. Modifique la primera línea de código en

```
print("Fundamentos","Programación","en")
print("Python")
```

utilizando las palabras clave `sep` y `end`, para que coincida con el resultado esperado:

```
Fundamentos***Programación***en...Python
```

Utilice dos funciones `print()` y no cambie nada en la segunda invocación de `print()`.

Exercise 1.2. Escriba una sola línea de código, utilizando la función `print()`, así como los caracteres de nueva línea y escape, para obtener la salida:

```
"Estoy"
""aprendiendo""
"""Python"""
```

Exercise 1.3. Millas y kilómetros son unidades de longitud o distancia. Una milla equivale aproximadamente a 1.61 kilómetros. Complete el programa:

```
kilometros = 12.25
millas = 7.38

millas_a_kilometros = ###
kilometros_a_millas = ###
```

```
print(millas, " millas son ", round(millas_a_kilometros, 2),\
" kilómetros ")
print(kilometros, " kilómetros son ", round(kilometros_a_millas, 2),\
" millas ")
```

para que convierta de:

- Millas a kilómetros.
- Kilómetros a millas.

No cambie el código existente. Escribe tu código en los lugares indicados con `###`. Prueba tu programa con los datos que han sido provistos en el código fuente. Resultado Esperado:

```
7.38 millas son 11.88 kilómetros
12.25 kilómetros son 7.61 millas
```

Exercise 1.4. Complete el código para evaluar y mostrar el resultado de cuatro operaciones aritméticas básicas:

```
# ingresa un valor flotante para la variable a aquí
# ingresa un valor flotante para la variable b aquí

# muestra el resultado de la suma aquí
# muestra el resultado de la resta aquí
# muestra el resultado de la multiplicación aquí
# muestra el resultado de la división aquí

print("\n¡Eso es todo, amigos!")
```

Quizá no puedas proteger el código de un usuario que intente dividir entre cero. Por ahora, no hay que preocuparse por ello.

Exercise 1.5. Complete el código para poder evaluar la siguiente expresión:

$$\frac{1}{x+\frac{1}{x+\frac{1}{x+\frac{1}{x}}}}.$$

El resultado debe de ser asignado a `y`. Se cauteloso, observa los operadores y priorízalos. Utiliza cuantos paréntesis sean necesarios. Puedes utilizar variables adicionales para acortar la expresión (sin embargo, no es muy necesario). Prueba tu código cuidadosamente.

```
x = float(input("Ingresa el valor para x: "))

# coloca tu código aquí

print("y =", y)
```

Exercise 1.6. *Espatifilo*, más comúnmente conocida como la planta de Cuna de Moisés o flor de la paz, es una de las plantas para interiores más populares que filtra las toxinas dañinas del aire. Imagina que tu programa de computadora ama estas plantas. Cada vez que recibe una entrada en forma de la palabra `Espatifilo`, grita involuntariamente a la consola la siguiente cadena: `"¡Espatifilo es la mejor planta de todas!"` Escribe un programa que utilice el concepto de ejecución condicional, tome una cadena como entrada y que:

- Imprima el enunciado `"Sí, ¡El Espatifilo es la mejor planta de todos los tiempos!"` en la pantalla si la cadena ingresada es `"Espatifilo"`.
- Imprima `"No, ¡quiero un gran Espatifilo!"` si la cadena ingresada es `"espatifilo"`.
- Imprima `"¡Espatifilo! ¡No [entrada]!"` de lo contrario. Nota: `[entrada]` es la cadena que se toma como entrada.

Prueba tu código con los datos que te proporcionamos. ¡Y hazte de un Espatifilo también!

Exercise 1.7. Como seguramente sabrás, debido a algunas razones astronómicas, el año pueden ser bisiesto o común . Los primeros tienen una duración de 366 días, mientras que los últimos tienen una duración de 365 días.

Desde la introducción del calendario gregoriano (en 1582), se utiliza la siguiente regla para determinar el tipo de año:

- Si el número del año no es divisible entre cuatro, es un año común.
- De lo contrario, si el número del año no es divisible entre 100, es un año bisiesto.
- De lo contrario, si el número del año no es divisible entre 400, es un año común.
- De lo contrario, es un año bisiesto.

Observa el código:

```
año = int(input("Introduzca un año:"))

# Coloca tu código aquí.
```

solo lee un número de año y debe completarse con las instrucciones que implementan la prueba que acabamos de describir.

El código debe mostrar uno de los dos mensajes posibles, que son `Año bisiesto` o `Año común`, según el valor ingresado. Sería bueno verificar si el año ingresado cae en la era gregoriana y emitir una advertencia de lo contrario: No dentro del período del calendario gregoriano.

Exercise 1.8. La sentencia `continue` se usa para omitir el bloque actual y avanzar a la siguiente iteración, sin ejecutar las declaraciones dentro del ciclo. Se puede usar tanto con `while` y ciclos `for`. Diseñe un devorador de vocales, para ello escribe un programa que use:

- Un ciclo `for`.
- El concepto de ejecución condicional (if-elif-else).
- La declaración `continue`.

Tu programa debe:

- Pedir al usuario que ingrese una palabra.
- Utilizar `userWord = userWord.upper()` para convertir la palabra ingresada por el usuario a mayúsculas; hablaremos sobre los llamados *métodos de cadena* y el `upper()` muy pronto, no te preocupes.
- Utilizar la ejecución condicional y la instrucción `continue` para "comer" las siguientes vocales A , E , I , O , U de la palabra ingresada.
- Imprimir las letras no consumidas en la pantalla, cada una de ellas en una línea separada.

Exercise 1.9. Tu tarea aquí es aún más especial que antes: ¡Debes rediseñar el devorador de vocales (feo) del ejercicio anterior y crear un mejor devorador de vocales (bonito) mejorado! Escribe un programa que use:

- Un ciclo `for`.
- El concepto de ejecución condicional (if-elif-else).
- La declaración `continue`.

Tu programa debe:

- Pedir al usuario que ingrese una palabra.
- Utilizar `userWord = userWord.upper()` para convertir la palabra ingresada por el usuario a mayúsculas; hablaremos sobre los llamados *métodos de cadena* y el `upper()` muy pronto, no te preocupes.
- Usa la ejecución condicional y la instrucción `continue` para "comer" las siguientes vocales A , E , I , O , U de la palabra ingresada.
- Asigne las letras no consumidas a la variable `palabrasinVocal` e imprime la variable en la pantalla.

Exercise 1.10. En 1937, un matemático alemán llamado Lothar Collatz formuló una hipótesis intrigante (aún no se ha comprobado) que se puede describir de la siguiente manera:

1. Toma cualquier número entero que no sea negativo y que no sea cero y asígnale el nombre `num`.

2. Si es par, evalúa un nuevo `num` como `num//2`.

3. De lo contrario, si es impar, evalúe un nuevo `num` como `3*num + 1`.

4. Si `num` es diferente de 1, salta al punto 2.

La conjetura dice que, independientemente del valor inicial de `num`, el valor siempre tiende a 1. Por supuesto, es una tarea extremadamente compleja usar una computadora para probar la conjetura de cualquier número natural, pero puede usar Python para verificar algunos números individuales.

Escribe un programa que lea un número natural y ejecute los pasos anteriores siempre que `num` sea diferente de 1. También queremos que cuente los pasos necesarios para lograr el objetivo. Tu código también debe mostrar todos los valores intermedios de `num`.

Exercise 1.11. Complete el siguiente código:

```
lista = [1, 2, 3, 4, 5]

# Paso 1: escribe una línea de código que solicite al usuario
# para reemplazar el número de en medio con un número entero ingresado por el usuario.

# Paso 2: escribe aquí una línea de código que elimine el último elemento de la lista.

# Paso 3: escribe aquí una línea de código que imprima la longitud de la lista existente.

print(listaSombrero)
```

Exercise 1.12. Los Beatles fueron uno de los grupos de música más populares de la década de 1960 y la banda más vendida en la historia. Algunas personas los consideran el acto más influyente de la era del rock. De hecho, se incluyeron en la compilación de la revista Time de las 100 personas más influyentes del siglo XX.

La banda sufrió muchos cambios de formación, que culminaron en 1962 con la formación de John Lennon, Paul McCartney, George Harrison y Richard Starkey (mejor conocido como Ringo Starr).

Escribe un programa que refleje estos cambios y le permita practicar con el concepto de listas. Tu tarea es:

Paso 1: Crea una lista vacía llamada `beatles`.

Paso 2: Emplea el método `append()` para agregar los siguientes miembros de la banda a la lista: `John Lennon`, `Paul McCartney` y `George Harrison`.

Paso 3: Emplea el ciclofor y el `append()` para pedirle al usuario que agregue los siguientes miembros de la banda a la lista: `Stu Sutcliffe`, y `Pete Best`.

Paso 4: Usa la instrucción `del` para eliminar a `Stu Sutcliffe` y `Pete Best` de la lista.

Paso 5: Usa el método `insert()` para agregar a `Ringo Starr` al principio de la lista.

Exercise 1.13. Imagina una lista: no muy larga ni muy complicada, solo una lista simple que contiene algunos números enteros. Algunos de estos números pueden estar repetidos, y esta es la clave. No queremos ninguna repetición. Queremos que sean eliminados.

Tu tarea es escribir un programa que elimine todas las repeticiones de números de la lista. El objetivo es tener una lista en la que todos los números aparezcan no más de una vez. Asume que la lista original está ya dentro del código, no tienes que ingresarla desde el teclado.

Exercise 1.14. Escribe y pruebe una función que toma un argumento (un año) y devuelve `True` si el año es un año bisiesto, o `False` sí no lo es. Complete:

```
def is_year_leap(year):
#
# coloca tu código aquí
#

test_data = [1900, 2000, 2016, 1987]
test_results = [False, True, True, False]
for i in range(len(test_data)):
   yr = test_data[i]
   print(yr,"->",end="")
   result = is_year_leap(yr)
   if result == test_results[i]:
      print("OK")
   else:
      print("Error")
```

Exercise 1.15. Escribe una función que verifique si un número es primo o no. La función:

- Se llama `is_prime`.
- Toma un argumento (el valor a verificar).
- Devuelve `True` si el argumento es un número primo, y `False` de lo contrario.

Completa:

```
def is_prime(num):
#
# coloca tu código aquí
#

for i in range(1, 20):
   if is_prime(i + 1):
      print(i + 1, end=" ")
print()
```

Exercise 1.16. Escribe una función capaz de ingresar valores enteros y verificar si están dentro de un rango especificado. La función debe:

- Aceptar tres argumentos: una entrada, un límite inferior aceptable y un límite superior aceptable.
- Si el usuario ingresa una cadena que no es un valor entero, la función debe emitir el mensaje `Error: entrada incorrecta`, y solicitará al usuario que ingrese el valor nuevamente.
- Si el usuario ingresa un número que está fuera del rango especificado, la función debe emitir el mensaje
  `Error: el valor no está dentro del rango permitido (min..max)`
  y solicitará al usuario que ingrese el valor nuevamente.
- Si el valor de entrada es válido, será regresado como resultado.

Prueba tu código cuidadosamente:

```
def readint(prompt, min, max):
#
# tu codigo aqui
#

v = readint("Ingresa un numero de -10 a 10: ", -10, 10)

print("El numero es:", v)
```

Exercise 1.17. Escribe tu propia función, que se comporte casi como el método original `split()`, por ejemplo:

- Debe aceptar únicamente un argumento: una cadena.
- Debe devolver una lista de palabras creadas a partir de la cadena, dividida en los lugares donde la cadena contiene espacios en blanco.
- Si la cadena está vacía, la función debería devolver una lista vacía.

- Su nombre debe ser `mi_split()`.

Prueba tu código con cuidado:

```
def mi_split(strng):
#
# coloca tu código aquí
#

print(mi_split("Ser o no ser, esa es la pregunta"))
print(mi_split("Ser o no ser,esa es la pregunta"))
print(mi_split("   "))
print(mi_split(" abc "))
print(mi_split(""))
```

Exercise 1.18. Un anagrama es una nueva palabra formada al reorganizar las letras de una palabra, usando todas las letras originales exactamente una vez. Por ejemplo, las frases "rail safety" y "fairy tales" son anagramas, mientras que "I am" y "You are" no lo son. Escribe un programa que:

- Le pida al usuario dos textos por separado.
- Compruebe si los textos ingresados son anagramas e imprima el resultado.

Nota:

- Supongamos que dos cadenas vacías no son anagramas.
- Tratar las letras mayúsculas y minúsculas como iguales.
- Los espacios no se toman en cuenta durante la verificación: trátalos como inexistentes.

Prueba tu código.

Exercise 1.19. Algunos dicen que el *Dígito de la Vida* es un dígito calculado usando el cumpleaños de alguien. Es simple: solo necesitas sumar todos los dígitos de la fecha. Si el resultado contiene más de un dígito, se debe repetir la suma hasta obtener exactamente un dígito. Por ejemplo:

- 1 Enero 2017 = 2017 01 01
- $2 + 0 + 1 + 7 + 0 + 1 + 0 + 1 = 12$
- $1 + 2 = 3$

3 es el dígito que buscamos y encontramos.

Tu tarea es escribir un programa que:

- Le pregunta al usuario su cumpleaños (en el formato AAAAMMDD o AAAADMM o MMDDAAAA; en realidad, el orden de los dígitos no importa).

- Da como salida El Dígito de la Vida para la fecha ingresada.

Prueba tu código.

Exercise 1.20. Vamos a jugar un juego. Le daremos dos cadenas: una es una palabra (por ejemplo, "dog") y la segunda es una combinación de un grupo de caracteres. Escribe un programa que responda la siguiente pregunta: ¿Los caracteres que comprenden la primera cadena están ocultos dentro de la segunda cadena? Por ejemplo:

- Si la segunda cadena es "vcxzxduybfdsobywuefgas", la respuesta es `sí`.
- Si la segunda cadena es "vcxzxdcybfdstbywuefsas", la respuesta es `no` (ya que no están las letras "d", "o", "g", ni en ese orden).

Sugerencia:

- Usa las variantes de dos argumentos de las funciones `pos()` dentro de tu código.
- No te preocupes por mayúsculas y minúsculas.

Prueba tu código.