

Comenzado el	domingo, 28 de mayo de 2023, 14:24
Estado	Finalizado
Finalizado en	domingo, 28 de mayo de 2023, 15:11
Tiempo empleado	47 minutos 23 segundos
Puntos	15/17
Calificación	9 de 10 (88%)

Pregunta **1**

Correcta

Se puntúa 1 sobre 1

En la columna de la izquierda se enumeran algunas situaciones típicas de programación. Seleccione de la columna de la derecha cuál es el tipo de ciclo que sería más adecuado o cómodo para usar en cada caso (sin que esto implique que el ciclo elegido sea obligatorio para cada caso...):

Programas controlados por menú	✓	Ciclo do while (forzado a la forma [1, N] y conteniendo un if anidado para chequear la opción ingresada) ▾
Recorrido de una secuencia (tupla, range, cadena, lista, etc.)	✓	Ciclo for (iterando sobre la estructura de datos a procesar) ▾
Esquema en el cual se sabe de antemano el número de repeticiones	✓	Ciclo for (iterando sobre un range cuyo tamaño coincida con la cantidad de repeticiones) ▾
Validación de valores cargados por teclado.	✓	Ciclo while (forzado a la forma [1, N] y que incluya una condición de refuerzo de error) ▾
Esquema en el cual se desconoce de antemano el número de repeticiones	✓	Ciclo while (operando en forma natural como [0, n]) ▾

¡Ok!

La respuesta correcta es: Programas controlados por menú → Ciclo do while (forzado a la forma [1, N] y conteniendo un if anidado para chequear la opción ingresada), Recorrido de una secuencia (tupla, range, cadena, lista, etc.) → Ciclo for (iterando sobre la estructura de datos a procesar), Esquema en el cual se sabe de antemano el número de repeticiones → Ciclo for (iterando sobre un range cuyo tamaño coincida con la cantidad de repeticiones), Validación de valores cargados por teclado. → Ciclo while (forzado a la forma [1, N] y que incluya una condición de refuerzo de error), Esquema en el cual se desconoce de antemano el número de repeticiones → Ciclo while (operando en forma natural como [0, n])

Pregunta **2**

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes son **ciertas** en cuanto al uso de **else** cuando acompaña a un ciclo en Python?

Seleccione una o más de una:

- ☒ a. Tanto en un *while* como en un *for*, la rama *else* no será ejecutada si el ciclo terminó por acción de una instrucción *break*. ✓ ¡Ok!
- ☐ b. Tanto en un *while* como en un *for*, la rama *else* no será ejecutada si el ciclo incluía una instrucción *continue* en su bloque de acciones.
- ☒ c. En un ciclo *for*, las instrucciones de la rama *else* se ejecutan cuando el *for* termina de iterar sobre *todos* los elementos de la colección o secuencia dada. ✓ ¡Ok!
- ☒ d. En un ciclo *while*, las instrucciones de la rama *else* se ejecutan en el momento en que la expresión de control del ciclo se evalúa en *False* (sin importar en qué vuelta se obtuvo el *False*). ✓ ¡Ok!

¡Correcto!

Las respuestas correctas son:

En un ciclo *while*, las instrucciones de la rama *else* se ejecutan en el momento en que la expresión de control del ciclo se evalúa en *False* (sin importar en qué vuelta se obtuvo el *False*),.

En un ciclo *for*, las instrucciones de la rama *else* se ejecutan cuando el *for* termina de iterar sobre *todos* los elementos de la colección o secuencia dada.,

Tanto en un *while* como en un *for*, la rama *else* no será ejecutada si el ciclo terminó por acción de una instrucción *break*.

Pregunta **3**

Correcta

Se puntúa 2 sobre 2

Suponga que se desea cargar por teclado un número, pero validando que no sea negativo. Se proponen los siguientes dos scripts para ello:

Script 1.)

```
n = -1
while n < 0:
    n = int(input('Ingrese un número no negativo: '))
    if n < 0:
        print('Se pidió no negativo... cargue otra vez...')
    else:
        print('El dato cargado fue validado en forma correcta...')
```

Script 2.)

```
n = -1
while n < 0:
    n = int(input('Ingrese un número no negativo: '))
    if n < 0:
        print('Se pidió no negativo... cargue otra vez...')
    else:
        print('El dato cargado fue validado en forma correcta...')
```

¿Cuál de las siguientes afirmaciones es correcta en relación a estos dos scripts?

Seleccione una:

- ☐ a. El script 1 es incorrecto (mostrará el mensaje "El dato cargado fue validado en forma correcta..." muchas veces) y el segundo es correcto.
- ☐ b. El script 1 es correcto, pero el segundo no lo es (nunca mostrará el mensaje "El dato cargado fue validado en forma correcta...")
- ☐ c. Ambos son incorrectos.
- ☒ d. Ambos son correctos. En la situación planteada, ambos hacen exactamente lo mismo. ✓ ¡Correcto!

¡Correcto!

La respuesta correcta es:

Ambos son correctos. En la situación planteada, ambos hacen exactamente lo mismo.

Pregunta **4**

Correcta

Se puntúa 1 sobre 1

Analice el siguiente programa básico controlado por un menú de opciones. ¿Hay algún error en el planteo del mismo?

```
__author__ = 'Catedra de AED'

op = 1
while op != 3:
    # visualizacion de las opciones...
    print('1. Opcion 1')
    print('2. Opcion 2')
    print('3. Opcion 3')
    print('4. Salir')
    op = int(input('Ingrese el numero de la opcion elegida: '))

    # chequeo de la opcion elegida...
    if op == 1:
        print('Eligió la opcion 1...')
    elif op == 2:
        print('Eligió la opcion 2...')
    elif op == 3:
        print('Eligió la opcion 3...')
```

Seleccione una:

- ☐ a. Sí. El error es que dentro del bloque de acciones del ciclo, no hay una condición que controle si *op* es 4, por lo que si se ingresa un 4 el programa se interrumpirá con un mensaje de error.
- ☒ b. Sí. El error es que en la lista de opciones la opción de salida está marcada con el número 4, pero el ciclo corta ✔ **¡Correcto!** cuando se ingresa la 3.
- ☐ c. Sí. El error es que el ciclo no controla si el valor ingresado en *op* es un número menor a 1 o mayor a 4, lo cual hace que si se carga un número incorrecto, el programa se interrumpirá con un mensaje de error.
- ☐ d. No. No hay ningún error en el programa.
- ☐ e. Sí. El error es que el ciclo no llega a hacer ninguna ejecución del bloque de acciones, ya la variable *op* comienza valiendo 1 y en ese momento la condición de control de ciclo se hace falsa.

¡Correcto!

La respuesta correcta es:

Sí. El error es que en la lista de opciones la opción de salida está marcada con el número 4, pero el ciclo corta cuando se ingresa la 3.

Pregunta 5

Correcta

Se puntúa 2 sobre 2

Considere el programa para el *Juego del Número Secreto* que se presentó en la Ficha 8. En ese programa se usa una bandera para marcar en el algoritmo si el número secreto fue encontrado o no. Nos proponemos tratar de eliminar el uso de esa bandera y simplificar la estructura del programa, y sugerimos el que se muestra más abajo emplando una *instrucción break para cortar el ciclo apenas se encuentre el número secreto*. ¿Funciona correctamente el programa que estamos sugiriendo? *Seleccione la respuesta que mejor describa lo que está pasando con el programa propuesto.*

```
__author__ = 'Catedra de AED'

import random

print('Juego del Número Secreto... Configuración Inicial...')
limite_derecho = int(input('El número secreto estará entre 1 y: '))
cantidad_intentos = int(input('Cantidad máxima de intentos que tendrá disponible: '))

# limites iniciales del intervalo de búsqueda...
izq, der = 1, limite_derecho

# contador de intentos...
intentos = 0

# el numero secreto...
secreto = random.randint(1, limite_derecho)

# el ciclo principal... siga mientras no
# haya sido encontrado el número, y la
# cantidad de intentos no llegue a 5...
while intentos < cantidad_intentos:
    intentos += 1
    print('\nEl numero está entre', izq, 'y', der)

    # un valor para forzar al ciclo a ser [1, N]...
    num = izq - 1


    # carga y validación del número sugerido por el usuario...
    while num < izq or num > der:
        num = int(input('[Intento: ' + str(intentos) + '] => Ingrese su numero: '))
        if num < izq or num > der:
            print('Error... le dije entre', izq, 'y', der, '...')

    # controlar si num es correcto, avisar y cortar el ciclo...
    if num == secreto:
        print('\nGenio!!! Acertaste en', intentos, 'intentos')
        break

    # ... pero si no lo es, ajustar los límites
    # del intervalo de búsqueda... y seguir...
    elif num > secreto:
        der = num
    else:
        izq = num

print('\nLo siento!!! Se te acabaron los intentos. El número era:', secreto)
```

Seleccione una:

- ☒ a. No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto se mostrará  ¡Correcto! correctamente el mensaje avisando que ganó y cortará el ciclo con la instrucción *break*. Pero como *break* corta el ciclo y no el programa completo, entonces el programa continuará e inmediatamente mostrará también el mensaje avisando que el jugador perdió, provocando ambigüedad.
- ☐ b. No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto se mostrará el mensaje avisándole que ganó pero el ciclo continuará pidiendo que se ingrese un número, hasta agotar el número de intentos disponible.
- ☐ c. Sí. Funciona correctamente para todos los casos.

- ☐ d. No. No funciona bien: en la forma en que está planteado, se muestran en forma incorrecta los mensajes informando los límites del intervalo que contiene al número secreto en cada vuelta del ciclo, ya que las variables *izq* y *der* se actualizan en forma incorrecta.

¡Correcto!

La respuesta correcta es:

No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto se mostrará correctamente el mensaje avisando que ganó y cortará el ciclo con la instrucción *break*. Pero como *break* corta el ciclo y no el programa completo, entonces el programa continuará e inmediatamente mostrará también el mensaje avisando que el jugador perdió, provocando ambigüedad.

Pregunta **6**

Correcta

Se puntúa 3 sobre 3

Considere la implementación del juego Piedra, Papel y Tijera que se analizó en la Ficha 8, y en particular la siguiente modificación del mecanismo para leer por teclado la jugada del participante humano. ¿Hay algún problema con esta variante? (Suponga que el usuario efectivamente cargará por teclado un número entre 1 y 3 para elegir la figura que desea jugar)

Observación: Puede haber más de una respuesta válida. Marque todas las que considere correctas.

```
# asuma que la variable descripcion es una variable global
# correctamente definida como una tupla de esta forma:
# descripcion = 'Piedra', 'Papel', 'Tijera'

humano = int(input('Ingresa 1 - Piedra, 2 - Papel o 3 - Tijera: '))
print('Usted eligió:', descripcion[humano])
```

Seleccione una o más de una:

- ☐ a. Si el usuario carga un 1 (Piedra) o un 2 (Papel) como figura elegida, se mostrará incorrectamente el nombre de la figura: en lugar de 'Piedra', le mostrará 'Tijera', y en lugar de 'Papel' le mostrará 'Piedra'.
- ☒ b. Si el usuario carga un 1 (Piedra) o un 2 (Papel) como figura elegida, se mostrará incorrectamente el nombre de la figura: en lugar de 'Piedra', le mostrará 'Papel', y en lugar de 'Papel' le mostrará 'Tijera'. ¡Ok! Esto se debe a que al no restar 1 al índice cargado por el usuario, la función está entrando incorrectamente al casillero de la tupla que contiene a cada descripción.
- ☐ c. No hay ningún problema.
- ☒ d. El proceso provocará un error y el programa se interrumpirá cuando el usuario cargue un 3 como figura elegida (la tupla `descripcion` no tiene un casillero con índice 3). ¡Ok!

¡Correcto!

Las respuestas correctas son:

El proceso provocará un error y el programa se interrumpirá cuando el usuario cargue un 3 como figura elegida (la tupla `descripcion` no tiene un casillero con índice 3),

Si el usuario carga un 1 (Piedra) o un 2 (Papel) como figura elegida, se mostrará incorrectamente el nombre de la figura: en lugar de 'Piedra', le mostrará 'Papel', y en lugar de 'Papel' le mostrará 'Tijera'.

Pregunta 7

Correcta

Se puntúa 3 sobre 3

Considere la implementación del juego Piedra, Papel y Tijera que se analizó en la Ficha 8, y en particular los tres procesos para *leer la jugada del humano, generar la jugada del computador y buscar si hubo un ganador* que originalmente contenía el programa:

```
# asuma que la variable descripcion es una variable global
# correctamente definida como una tupla de esta forma:
# descripcion = 'Piedra', 'Papel', 'Tijera'

# leer jugada del humano...
humano = int(input('Ingrese 1 - Piedra, 2 - Papel o 3 - Tijera: '))
print('Usted eligio:', descripcion[humano - 1])

# generar jugada del computador...
computadora = random.randint(1, 3)
print('La computadora eligio:', descripcion[computadora - 1])


# determinar si hubo un ganador...
if humano != computadora:
    if (humano == 1 and computadora == 3) \
        or (humano == 3 and computadora == 2) \
        or (humano == 2 and computadora == 1):
        ganador = 1
    else:
        ganador = -1
else:
    ganador = 0
```

Suponga que el usuario efectivamente cargará por teclado un número entre 1 y 3 para elegir la figura que desea jugar. ¿Qué efecto causaría en el programa que *el proceso para generar la jugada del computador* fuese reemplazado por esta otra versión que se muestra a continuación?

```
# asuma que la variable descripcion es una variable global
# correctamente definida como una tupla de esta forma:
# descripcion = 'Piedra', 'Papel', 'Tijera'

computadora = random.randint(0, 2)
print('La computadora eligio:', descripcion[computadora])
```

Seleccione una:

- ☐ a. La nueva versión ejecuta sin interrumpirse ni lanzar un error, pero funciona mal: al seleccionar en forma aleatoria un número entre 0 y 2, mostrará mal las descripciones de las figuras seleccionadas.
- ☐ b. No causará ningún problema. El programa seguirá funcionando y mostrando sus resultados en forma correcta.
- ☒ c. En si misma la nueva versión funciona bien, pero el número seleccionado de cada figura estará entre 0 y 2, haciendo que  ¡Ok! *el proceso para buscar un ganador* falle en su lógica ya que el humano está cargando valores entre 1 y 3.
- ☐ d. La nueva versión se interrumpirá al ejecutarla, lanzando un mensaje de error, si el número seleccionado por *randint()* es 0.

¡Correcto!

La respuesta correcta es:

En si misma la nueva versión funciona bien, pero el número seleccionado de cada figura estará entre 0 y 2, haciendo que *el proceso para buscar un ganador* falle en su lógica ya que el humano está cargando valores entre 1 y 3.

Pregunta 8

Incorrecta

Se puntúa 0 sobre 2

Suponga que se quiere cargar por teclado el valor una temperatura, validando que la misma esté entre -70 grados y 50 grados, y se plantea un ciclo *while* forzado a operar en forma $[1, N]$, como muestra el esquema, para hacerlo:

```
__author__ = 'Catedra de AED'

t = 51
while ????? :
    t = int(input('Cargue temperatura (entre -70 y 50, por favor): '))
```

¿Cuál de las siguientes debería ser la condición a incluir en ese ciclo *while* (en el lugar donde figuran los signos de pregunta de color rojo) para que el *valor vuelva a cargarse* en caso de haber sido mal ingresado?

Seleccione una:

- ☒ a. $t < -70$ and $t > 50$ ✖ Incorrecto... ningún número t puede hacer que esa condición sea cierta...
- ☐ b. $t \geq -70$ and $t \leq 50$
- ☐ c. $t < -70$ or $t > 50$
- ☐ d. $t \geq -70$ or $t \leq 50$

Revise la Ficha 7, páginas 163 y 164.

La respuesta correcta es:

 $t < -70$ or $t > 50$

Pregunta 9

Correcta

Se puntúa 1 sobre 1

Cuáles de las siguientes afirmaciones son ciertas en cuanto al uso de ciclos o instrucciones repetitivas en Python? (Aclaración: más de una respuesta puede ser correcta... marque TODAS las que considere válidas...)

Seleccione una o más de una:

- ☐ a. Todos los ciclos básicos de Python, son de la forma $[1-N]$.
- ☒ b. Todos los ciclos básicos de Python, son de la forma $[0, N]$. ✔ ¡Ok!
- ☒ c. Python provee dos tipos de ciclos básicos: el *while* y el *for*. ✔ ¡Ok!
- ☐ d. El ciclo *while* de Python **sólo puede aplicarse** cuando se desconoce la cantidad de repeticiones a realizar. Si la cantidad de repeticiones se conoce de antemano, **debe** aplicarse el *for*.

¡Correcto!

Las respuestas correctas son:

Python provee dos tipos de ciclos básicos: el *while* y el *for*.Todos los ciclos básicos de Python, son de la forma $[0, N]$.


Pregunta **10**

Correcta

Se puntúa 1 sobre 1

¿Qué se entiende por **momento epoch** cuando se trabaja con ciertas funciones para medir tiempos en Python, y cuál es ese momento?

Seleccione una:

- ☐ a. Al momento (arbitrariamente elegido) a partir del cual se mide el tiempo transcurrido en Python. Todos los sistemas operativos usan exactamente el mismo momento (misma fecha y hora) de inicio del tiempo.
- ☒ b. Al momento (arbitrariamente elegido) a partir del cual se mide el tiempo transcurrido en Python. De acuerdo a cada  ¡Ok!
sistema operativo, ese momento inicial del tiempo puede ser establecido en fechas y horas diferentes.
- ☐ c. Al momento a partir del cual se mide el tiempo transcurrido en Python. Ese momento coincide con la fecha y hora en la cual Python fue instalado en el computador del programador que mide el tiempo.
- ☐ d. Al momento en el cual la licencia de uso libre de Python vencerá. Ese momento es exactamente el día 31/12/2030, a las 23:59.

¡Correcto!

La respuesta correcta es:

Al momento (arbitrariamente elegido) a partir del cual se mide el tiempo transcurrido en Python. De acuerdo a cada sistema operativo, ese momento inicial del tiempo puede ser establecido en fechas y horas diferentes.

[◀ Ficha Opcional 08 \[Actualización de datos\] \(PDF - para lectura directa\)](#)

Ir a...



[Guía de Ejercicios Prácticos - Ficha 08 ▶](#)