Comenzado el	jueves, 23 de agosto de 2018, 21:34
Estado	Finalizado
Finalizado en	jueves, 23 de agosto de 2018, 21:49
Tiempo empleado	15 minutos 33 segundos
Puntos	15/15
Calificación	**10** de 10 (**100**%)

Pregunta 1

Correcta

Puntúa 2 sobre 2

Suponga que se han creado y cargado por teclado correctamente dos arreglos paralelos llamados *numeros* y *codigos* en los que se han almacenado datos de socios de un club deportivo. En el primer arrreglo se guardaron los números de identificación de los socios, y en el segundo se guardaron números entre 0 y 9 que identifican el deporte que cada socio practica. Se presenta más abajo una función *count()* que lleva a cabo un conteo para determinar cuántos socios están registrados en cada uno de los 10 deportes posibles, asumiendo que los códigos que se cargan en el arreglo *codigos* están validados en el momento de hacer la carga, para garantizar que efectivamente sean todos números entre 0 y 9 ¿Qué puede decirse que es cierto respecto de esta función si ***NO*** se tuviese la garantía de que esos códigos estén validados en la carga? (Más de una respuesta puede ser cierta... marque TODAS las que considere aplicables)

```
def count(codigos):
    vc = 10 * [0]
    for d in codigos:
        vc[d] += 1

    print('Cantidad de socios en cada deporte
 disponible:')
    for i in range(10):
        if vc[i] != 0:
            print('Codigo de deporte:', i, 'Ca
ntidad de socios registrados:', vc[i])
```

Seleccione una o más de una:

- [ ] a.
  Si aparece un código negativo o mayor a 9, el programa no se interrumpirá, pero finalmente el vector de conteos quedará con todos sus casilleros valiendo None (el proceso completo quedará invalidado, pero el programa continuará)

- [ ] b.
  No habría ningún problema: la función haría su trabajo correctamente de todos modos, sea cual sea el valor del código de deporte que ingrese.

- [ ] c.
  El programa no se interrumpirá, pero por cada código que aparezca cuyo valor sea mayor a 9, se agregarán casilleros al vector de conteo en forma automática, mostrando finalmente un conteo de más de 10 deportes.

- [x] d.
  Podría ocurrir que entre un codigo que no esté entre 0 y 9, provocando que la instrucción vc[d] += 1 lance un error de índice fuera de rango y el programa se interrumpa. ✓

  ¡Correcto!

¡Correcto!

La respuesta correcta es:
Podría ocurrir que entre un codigo que no esté entre 0 y 9, provocando que la instrucción vc[d] += 1 lance un error de índice fuera de rango y el programa se interrumpa.

Pregunta 2

Correcta

Puntúa 2 sobre 2

Otra vez suponga que se han creado y cargado por teclado correctamente dos arreglos paralelos llamados *numeros* y *codigos* en los que se han almacenado datos de socios de un club deportivo. En el primer arrreglo se guardaron los números de identificación de los socios, y en el segundo se guardaron números entre 0 y 9 que identifican el deporte que cada socio practica. Se presenta más abajo una función *count()* que lleva a cabo un conteo para determinar cuántos socios están registrados en cada uno de los 10 deportes posibles, asumiendo que los códigos que se cargan en el arreglo *codigos* están validados en el momento de hacer la carga, para garantizar que efectivamente sean todos números entre 0 y 9. Suponga ahora que en esa función, la creación del vector de conteo *vc* se hace en la forma que indica más abajo en color rojo. ¿Qué puede decirse respecto de la forma en que afecta al programa este replanteo de la función?

```
def count(codigos):
    vc = 10 * [None]
    for d in codigos:
        vc[d] += 1

    print('Cantidad de socios en cada deporte
 disponible:')
    for i in range(10):
        if vc[i] != 0:
            print('Codigo de deporte:', i, 'Ca
ntidad de socios:', vc[i])
```

Seleccione una:

a.
La función hará el conteo pero en forma incorrecta: todos los contadores quedarán finalmente valiendo None, aunque el programa no se interrrumpirá.

b.
La función hará el conteo pero en el arreglo sobra un casillero: si los posibles códigos de los deportes disponibles van entre 0 y 9, entonces la inicialización debió ser de la forma vc = 9 * [None], pues de otro modo estaría de más el casillero vc[10].

c.
La función provocará un fallo y el programa se interrumpirá: si cada casillero del arreglo *vc* tiene valor inicial *None*, cuando se pretenda acceder a un casillero para incrementar en 1 su valor se producirá el fallo. ✓

¡Correcto!

d.
La función hace su trabajo en forma normal y correctamente. El conteo de los socios por cada deporte se hará sin problemas.

¡Correcto!

La respuesta correcta es:
La función provocará un fallo y el programa se interrumpirá: si cada casillero del arreglo *vc* tiene valor inicial *None*, cuando se pretenda acceder a un casillero para incrementar en 1 su valor se producirá el fallo.

Pregunta 3

Correcta

Puntúa 2 sobre 2

Suponga que se ha creado y cargado por teclado un arreglo *peaje* en el que almacenaron cadenas de caracteres que representan las patentes de los vehículos que se registraron en una estación de peaje en un período dado. Suponga que se pide desarrollar una función que determine cuantas veces está registrada en el arreglo *peaje* la patente cargada en la variable *pat*. Suponga que se propone para cumplir esa tarea la función *search()* que se muestra más abajo, con la rama else que se ve en rojo como parte del if dentro del ciclo. ¿Qué puede decirse respecto del planteo de esa función *search()*?

```
def search(peaje, pat):
    c = 0
    for p in peaje:
        if p == pat:
            c += 1
        else:
            print('No esta registrado ese vehiculo')

    if(c != 0):
        print('Cantidad de pasos registrados para ese vehiculo:', c)
```

Seleccione una:

a.
El programa funcionará correctam*ente* sin ningún inconveniente.

b.
La función sólo contará una vez la patente indicada por *pat.*

c.
El planteo ***es incorrecto***: cada vez que se compruebe una casilla que NO tenga la patente buscada, aparecerá el mensaje indicando que no existe (aún cuando esa patente podría existir una o muchas veces). ✓

¡Correcto!

d.
La combinación entre el *else* y el contador *c* que aparece dentro del ciclo, provoca que la función cuente sólo algunas de las patentes cuyo valor coincida con *pat*: sólo aquellos que se encuentren en casillas con índice par (los de las casillas 0, 2, 4, 6, etc.)

¡Correcto!

La respuesta correcta es:
El planteo ***es incorrecto***: cada vez que se compruebe una casilla que NO tenga la patente buscada, aparecerá el mensaje indicando que no existe (aún cuando esa patente podría existir una o muchas veces).

Pregunta 4

Correcta

Puntúa 2 sobre 2

Otra vez suponga que se ha creado y cargado por teclado un arreglo *peaje* en el que almacenaron cadenas de caracteres que representan las patentes de los vehículos que se registraron en una estación de peaje en un período dado. Suponga que se pide desarrollar una función que determine cuantas veces está registrada en el arreglo *peaje* la patente cargada en la variable *pat*. Suponga que se propone para cumplir esa tarea la función *search()* que se muestra más abajo, con la instrucción break que se ve en rojo al final de la rama vervadera de if dentro del ciclo ¿Qué puede decirse respecto del planteo de esa función *search()*?

```
def search(peaje, pat):
    c = 0
    for p in peaje:
        if p == pat:
            c += 1
            break

    if(c != 0):
        print('Cantidad de pasos registrados
 para ese vehiculo:', c)
    else:
        print('No esta registrado ese vehicul
o')
```

Seleccione una:

a.
El programa funcionará correctamente de todos modos, y contará todas las repeticiones del valor *pat*.

b.
La función ya no cumplirá con el objetivo: si el vector peaje tuviese efectivamente más de una vez un valor igual a *pat*, contaría sólo el primero de ellos y detendría el ciclo. Así planteada, la función está haciendo una búsqueda secuencial de primera ocurrencia, en lugar de un conteo de todas las ocurrencias. ✓

¡Correcto!

c.
La combinación entre la instrucción *break* y el contador *c* que aparece dentro del ciclo, provoca que la función cuente sólo algunas de las ocurrencias del valor *pat*: sólo aquellas que se encuentren en casillas con índice par (los de las casillas 0, 2, 4, 6, etc.)

d.
La función contará todas las ocurrencias del valor *pat*, salvo la primera.

¡Correcto!

La respuesta correcta es:
La función ya no cumplirá con el objetivo: si el vector peaje tuviese efectivamente más de una vez un valor igual a *pat*, contaría sólo el primero de ellos y detendría el ciclo. Así planteada, la función está haciendo una búsqueda secuencial de primera ocurrencia, en lugar de un conteo de todas las ocurrencias.

Pregunta 5

Correcta

Puntúa 2 sobre 2

Suponga que se han creado y cargado por teclado dos arreglos *patentes* y *cabinas* para almacenar datos de los vehículos que se registraron en las distintas cabinas de una estación de peaje en un período dado. En el arreglo *patentes* se almacenaron cadenas que representan los números de patente de cada vehículo, y el arreglo *cabinas* se guardaron los números de las cabinas por las que cada vehículo pasó. Suponga que se pide una función *display(patentes, cabinas, x)*, que muestre sólo las patentes de los vehículos que hayan pasado por la cabina *x*.Una forma correcta de hacerlo sería la que se muestra ahora:

```
# version correcta...
def display(patentes, cabinas, x):
    exists = False
    print('Listado de vehiculos que pasaron por la cabina', x,
 ':')
    for i in range(len(cabinas)):
        if cabinas[i] == x:
            exists = True
            print('Patente:', patentes[i])

    if not exists:
        print('No se registraron vehiculos que hayan pasado por e
sa cabina')
```

Suponga ahora que esa función se modifica en la forma en que se muestra más abajo, eliminando la variable *exists* que originalmente se usaba en la función. ¿Qué puede decirse respecto de la forma en que afecta al programa este replanteo de la función *display()*?

```
# version modificada y sugerida para este Cues
tionario...
def display(patentes, cabinas, x):
    print('Listado de vehiculos que pasaron por la cabina', x,
 ':')
    for i in range(len(cabinas)):
        if cabinas[i] == x:
            print('Patente:', patentes[i])
```

Seleccione una:

a.
La versión modificada no funciona correctamente: si no hay ningún registro para mostrar, el programa se interrumpe al finalizar el ciclo y lanza un mensaje de error en la consola.

b.
La versión modificada no funciona correctamente: muestra siempre todos los datos, en lugar de mostrar sólo los que pasaron por la cabina *x*.

c.

El funcionamiento es correcto de todos modos, con una pequeña diferencia de interfaz de usuario: en la versión original, si no había ningún vehículo que haya pasado por la cabina *x*, la función mostraba un mensaje aclarando que no había ninguno. Pero en la versión modificada, ese mensaje no aparece y el listado quedará en blanco, sólo con el título. ✓

¡Correcto!

d.
El programa funcionará correctamente sin ninguna diferencia.

¡Correcto!

La respuesta correcta es:
El funcionamiento es correcto de todos modos, con una pequeña diferencia de interfaz de usuario: en la versión original, si no había ningún vehículo que haya pasado por la cabina *x*, la función mostraba un mensaje aclarando que no había ninguno. Pero en la versión modificada, ese mensaje no aparece y el listado quedará en blanco, sólo con el título.

Pregunta 6

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes ***NO ES*** una de las técnicas básicas (y ya obsoletas) consideradas clásicas en el campo de la criptología?

Seleccione una:

- a. El cifrado de Vernam.
- b. El cifrado de Vigenère.
- c. El cifrado de César.
- d. El cifrado RSA de Clave Pública. ✓ ¡Correcto!

¡Correcto!

La respuesta correcta es: El cifrado RSA de Clave Pública.

Pregunta 7

Correcta

Puntúa 1 sobre 1

¿Cuál es el principal problema de la *encriptación de César*?

Seleccione una:

- a. Es muy difícil de implementar en un lenguaje de programación.
- b. Es fácil de romper: sólo deben probarse tantas posibilidades de desplazamiento como caracteres haya en el alfabeto del mensaje. ✓

  ¡Correcto!
- c. No hay ningún problema: es el método de encriptación más seguro que se conoce hasta hoy.
- d. Produce mensajes encriptados con pérdida de información.

¡Correcto!

La respuesta correcta es:
Es fácil de romper: sólo deben probarse tantas posibilidades de desplazamiento como caracteres haya en el alfabeto del mensaje.

Pregunta 8

Correcta

Puntúa 1 sobre 1

¿Cuál es el principal problema de la *encriptación por Tabla de Transposición*?

Seleccione una:

a.
No hay ningún problema. Es un método de encriptación seguro.

b.
La cantidad de tablas de transposición diferentes que tendría que analizar un atacante es pequeña, y por lo tanto el encriptado es simple de quebrar..

c.
La tabla de transposición puede no entrar en la memoria disponible, y hacer entonces imposible su implementación.

d.
Si bien es más seguro que el cifrado de César, un atacante podría deducir la relación entre los caracteres analizando grupos de letras o sabiendo las frecuencias de las letras más comunes en cada idioma.

¡Correcto!

¡Correcto!

La respuesta correcta es:
Si bien es más seguro que el cifrado de César, un atacante podría deducir la relación entre los caracteres analizando grupos de letras o sabiendo las frecuencias de las letras más comunes en cada idioma.

Pregunta 9

Correcta

Puntúa 1 sobre 1

¿Cuál (y por qué) es la salida que producirá en pantalla el siguiente script en el que se aplica el método *find()*?

```
cad = 'Hola mundo otra vez'
r = cad.find('un')
print('Posición:', r)
```

Seleccione una:

a.
La salida será: *Posición: True*, porque la cadena *cad* contiene a la cadena 'un' que se está buscando.

b.
La salida será: *Posición: 6*, porque 6 es el índice dentro de la cadena *cad* en el que ***termina*** la cadena 'un' que se está buscando.

c.
La salida será: *Posición: -1*, porque la cadena *cad* no contiene a la palabra 'un' que se está buscando.

d.
La salida será: *Posición: 6*, porque 6 es el índice dentro de la cadena *cad* en el que ***comienza*** la cadena 'un' que se está buscando.

¡Correcto!

¡Correcto!

La respuesta correcta es:
La salida será: *Posición: 6*, porque 6 es el índice dentro de la cadena *cad* en el que ***comienza*** la cadena 'un' que se está buscando.

Pregunta 10

Correcta

Puntúa 1 sobre 1

¿Cuál es la salida que producirá en pantalla el siguiente script en el que se aplica el método *join()*?

```
v = ['Hola', 'mundo', 'otra', 'vez']
r = '*-*'.join(v)
print(r)
```

Seleccione una:

a. `Hola**mundo**otra**vez`

b. `Hola*-*mundo*-*otra*-*vez` ✓

¡Correcto!

c. `Holamundootravez`

d. `Hola mundo otra vez`

¡Correcto!

La respuesta correcta es:
`Hola*-*mundo*-*otra*-*vez`