Comenzado el	jueves, 28 de julio de 2022, 08:34
Estado	Finalizado
Finalizado en	jueves, 28 de julio de 2022, 09:04
Tiempo empleado	30 minutos 33 segundos
Comentario -	**8(ocho)**

Pregunta **1**

Correcta

Puntúa como 1

¿Cuál de las siguientes es **incorrecta** en relación a las características de una función en Python?

Seleccione una:

a. El cuerpo de una función puede (o no) tener una instrucción *return*.

b. En la cabecera de una función en Python los parámetros siempre deben estar presentes. ✔ ¡Correcto! Esta definición no es cierta, que es lo que se pidió marcar...

c. La cabecera de la función es la primera línea de la misma, donde entre otros elementos se define su nombre.

d. En Python una función es un segmento de programa que se codifica en forma separada y con un identificador para poder activarla

¡Correcto!

La respuesta correcta es:
En la cabecera de una función en Python los parámetros siempre deben estar presentes.

Pregunta **2**

Correcta

Puntúa como 1

Suponga que dispone de cuatro algoritmos diferentes para resolver el mismo problema, y que se sabe que los tiempos de ejecución (en el peor caso) son, respectivamente: ***O(n*log(n))***, ***O(n^2)***, ***O(n^3)*** y ***O(n)***.

¿Cuál de esos tres algorimos debería elegir, suponiendo que todos hacen el mismo consumo razonable de memoria?

Seleccione una:

a. El algoritmo cuyo tiempo de ejecución es O(n^2)

b. El algoritmo cuyo tiempo de ejecución es O(n) ✔ ¡Correcto! Efectivamente, este algoritmo sería el más rápido...

c. El algoritmo cuyo tiempo de ejecución es O(n^3)

d. El algoritmo cuyo tiempo de ejecución es O(n*log(n))

¡Correcto!

La respuesta correcta es:
El algoritmo cuyo tiempo de ejecución es O(n)

Pregunta **3**

Correcta

Puntúa como 1

¿Cuál es el valor que termina valiendo la variable *res* luego de la siguiente secuencia, en la que se usa el operador *resto* o *módulo* de una división?

```
a = 17
b = 3
res = a % b
```

Respuesta: 2 ✔

¡Ok! El resto de la división entre 17 y 3 es, efectivamente, 2.

La respuesta correcta es: 2

Pregunta **4**

Correcta

Puntúa como 1

¿Cuál de las siguientes ***no es*** una forma de codificación de caracteres para el estándar *Unicode*?

Seleccione una:

a. UTF-16

b. EBCDIC ✔

¡Correcto! La codificación EBCDIC (por **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode) es un sistema de codificación de caracteres empleado por IBM en sus computadoras mainframe, diferente de ASCII y otros estándares que se impusieron en el mercado, pero no es una forma de codificación de caracteres para Unicode.

c. UTF-32

d. UTF-8

¡Correcto!

La respuesta correcta es:
EBCDIC

Pregunta **5**

Correcta

Puntúa como 1

¿Cuál de los siguientes es el inconveniente principal que supone procesar el *mismo archivo de texto* en *diferentes plataformas* (o sistemas operativos)?

Seleccione una:

a. No hay ningún inconveniente especial. El mismo archivo de texto puede abrirse y manejarse sin problema alguno en cualquier plataforma, sin tener que tomar precauciones particulares.

b. El tratamiento dado al *salto de linea*: en algunas plataformas se representa con un caracter de control simple (\n) y en otras con una pareja (\r\n). En general, los lenguajes de programación modernos (como Python) resuelven automáticamente este inconveniente. ¡Correcto!

c. La forma en que se trata y reconoce el final del archivo. Cada plataforma hace esto de forma diferente en un archivo de texto, y con cada lenguaje de programación debe considerarse este problema en forma particular.

d. La forma en que se distinguen las mayúsculas de las minúsculas dentro del archivo.

¡Correcto!

La respuesta correcta es:
El tratamiento dado al *salto de linea*: en algunas plataformas se representa con un caracter de control simple (\n) y en otras con una pareja (\r\n). En general, los lenguajes de programación modernos (como Python) resuelven automáticamente este inconveniente.

Pregunta **6**

Correcta

Puntúa como 1

¿Cuáles de los siguientes son *factores esenciales* a tener en cuenta cuando se realiza el *análisis de la eficiencia de un algoritmo* (por ejemplo, para efectuar una comparación de rendimiento entre dos algoritmos que resuelven el mismo problema)?

Seleccione una o más de una:

- [ ] a. El diseño de la interfaz de usuario.
- [x] b. La complejidad aparente del código fuente. ¡Correcto!
- [x] c. El tiempo de ejecución. ¡Correcto!
- [x] d. El consumo de memoria. ¡Correcto!

¡Correcto!

Las respuestas correctas son:
El tiempo de ejecución.,

El consumo de memoria.,

La complejidad aparente del código fuente.

Pregunta **7**

Correcta

Puntúa como 1

Suponga que para un problema dado se ha planteado un algoritmo basado en divide y vencerás cuya estructura lógica esencial es la siguiente:

```
proceso(partición):
  adicional() [O(n)]
  proceso(partición/3)
  proceso(partición/3)
  proceso(partición/3)
```

¿Cuál de las siguientes es la expresión en *notación Big O* del tiempo de ejecución para este proceso? (Aclaración: la base del logaritmo no es esencial en una expresión en notación Big O, pero en este caso es relevante a los efectos del planteo conceptual de la pregunta).

Seleccione una:

a. $O(n^2 * \log_2(n))$

b. $O(n * \log_2(n))$

c. $O(n * \log_3(n))$ ✔ ¡Correcto!

d. $O(n^2 * \log_3(n))$

¡Correcto!

La respuesta correcta es:
$O(n * \log_3(n))$

Pregunta **8**

Correcta

Puntúa como 1

El proceso de búsqueda del mayor en el siguiente programa, contiene una modificación de la primera variante analizada en la Ficha 7: En ambas se usa un ciclo for que ejecute *n* repeticiones para cargar los *n* números. Pero en la versión original de la Ficha, el ciclo se ajusta como *for i in range(1, n+1)* mientras que ahora proponemos *for i in range(n)* para intentar abreviar, basándonos en la idea de que en ambos casos el ciclo hará efectivamente *n* repeticiones. ¿Hay algún problema con este cambio propuesto?

```
__author__ = 'Catedra de AED'


print('Determinacion del mayor de una sucesion (for cambiado)...')

n = int(input('Numero: '))
for i in range(n):
    num = int(input('Numero: '))
    if i == 1:
        may = num
    elif num > may:
        may = num

print('El mayor es:', may)
```

Seleccione una:

a. No. No hay ningún problema. El funcionamiento es exactamente igual al de la versión original.

b. Sí, hay un problema: el rango generado con *range(n)* contiene efectivamente *n* números y el ciclo hará *n* repeticiones, pero el primer valor del rango será 0 (y no 1) con lo cual al preguntar si *i == 1* en la primera vuelta del ciclo se obtendrá un falso. La variable *may* quedará entonces sin definir, y en esa misma vuelta el programa se interrumpirá al comparar *num* con *may*. ¡Correcto!

c. Sí, hay un problema: el rango generado con *range(n)* contiene efectivamente *n* números y el ciclo hará *n* repeticiones, pero el primer valor del rango será 0 (y no 1). Por lo tanto, la condición i == 1 sólo verdadera en la segunda vuelta del ciclo y la variable may quedará inicializada con el segundo número en lugar del primero. El programa entonces, ignorará el primer número que se cargue, y mostrará el mayor de la secuencia pero sin incluir al primero.

d. Sí, hay un problema: el rango generado con *range(n)* no contiene efectivamente *n* números, sino *n-1* números y el ciclo hará entonces una repetición menos de las esperadas.

¡Correcto!

La respuesta correcta es:
Sí, hay un problema: el rango generado con *range(n)* contiene efectivamente *n* números y el ciclo hará *n* repeticiones, pero el primer valor del rango será 0 (y no 1) con lo cual al preguntar si *i == 1* en la primera vuelta del ciclo se obtendrá un falso. La variable *may* quedará entonces sin definir, y en esa misma vuelta el programa se interrumpirá al comparar *num* con *may*.

Pregunta **9**

Parcialmente correcta

Puntúa como 1

¿Cuáles de las siguientes afirmaciones son ***verdaderas*** respecto de la *descomposición en factores primos* (o *factorización*) de un número *n entero y positivo*? (Observación: más de una respuesta puede ser válida. Marque todas las que considere correctas)

Seleccione una o más de una:

- [x] a. La factorización de todo número *n* entero y positivo es única.

¡Correcto! La demostración de este hecho constituye hoy el Teorema Fundamental de la Aritmética, comprobado ya por Euclides.

- [ ] b. La factorización de un número *n* entero y positivo puede contener más de una vez al mismo factor primo.
- [ ] c. Ningún número entero impar $n > 2$ puede contener al 2 en su factorización.
- [ ] d. Ningún número entero impar $n > 2$ puede contener números impares en su factorización.

Ha seleccionado correctamente 1.
Las respuestas correctas son:
La factorización de todo número *n* entero y positivo es única.,

La factorización de un número *n* entero y positivo puede contener más de una vez al mismo factor primo.,

Ningún número entero impar $n > 2$ puede contener al 2 en su factorización.

Pregunta **10**

Parcialmente correcta

Puntúa como 1

¿Cuáles de las siguientes afirmaciones ***son ciertas*** en referencia a las *Estrategias de Resolución de Problemas* que se citan? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

- [ ] a. El empleo de la *Recursividad* para resolver un problema no es recomendable en ningún caso, debido a la gran cantidad de recursos de memoria o de tiempo de ejecución que implica.
- [x] b. La estrategia de *Backtracking* es de base recursiva y permite implementar soluciones de prueba y error explorando las distintas soluciones y voviendo atrás si se detecta que un camino conduce a una solución incorrecta. cuando es aplicable, es más eficiente que la Fuerza Bruta, ya que permite eliminar caminos por deducción. ¡Correcto!
- [ ] c. La técnica de Programación Dinámica se basa en calcular los resultados de los subproblemas de menor orden o tamaño que pudieran aparecer, almacenar esos resultados en una tabla, y luego re-usarlos cuando vuelvan a ser requeridos en el cálculo del problema mayor.
- [x] d. La estrategia de *Fuerza Bruta* se basa en aplicar ideas intuitivas y directas, simples de codificar, pero normalmente produce algoritmos de mal rendimiento en tiempo de ejecución y/o de espacio de memoria empleado. ¡Correcto!

Las respuestas correctas son:
La estrategia de *Fuerza Bruta* se basa en aplicar ideas intuitivas y directas, simples de codificar, pero normalmente produce algoritmos de mal rendimiento en tiempo de ejecución y/o de espacio de memoria empleado.,

La estrategia de *Backtracking* es de base recursiva y permite implementar soluciones de prueba y error explorando las distintas soluciones y voviendo atrás si se detecta que un camino conduce a una solución incorrecta. cuando es aplicable, es más eficiente que la Fuerza Bruta, ya que permite eliminar caminos por deducción.,

La técnica de Programación Dinámica se basa en calcular los resultados de los subproblemas de menor orden o tamaño que pudieran aparecer, almacenar esos resultados en una tabla, y luego re-usarlos cuando vuelvan a ser requeridos en el cálculo del problema mayor.

Pregunta **11**

Parcialmente correcta

Puntúa como 1

Suponga que *m* es una variable tipo *file object*, que representa a un archivo ya abierto (o que será abierto cuando corresponda). ¿A qué posición (interna o externa) del archivo quedará apuntando el *file pointer* de ese archivo luego de las siguientes invocaciones al método *seek()*? (Cuando lo necesite, *suponga que el tamaño del archivo es de 20 bytes* y calcule las respuestas a partir de este dato)

`m.seek(5, io.SEEK_END)`	Apuntará al byte número 15. ✗
`m.seek(10, io.SEEK_SET)` `m.seek(-3, io.SEEK_CUR)`	Apuntará al byte número 7. ✓
`m.seek(-5, io.SEEK_END)`	Apuntará al byte número 25. ✗
`m.seek(-5, io.SEEK_SET)`	Provocará un error de runtime (file pointer negativo). ✓
`m.seek(5, io.SEEK_SET)`	Apuntará al byte número 5. ✓

Ha seleccionado correctamente 3.

La respuesta correcta es:

```
m.seek(5, io.SEEK_END)
```

→ Apuntará al byte número 25.,

```
m.seek(10, io.SEEK_SET)
m.seek(-3, io.SEEK_CUR)
```

→ Apuntará al byte número 7.,

```
m.seek(-5, io.SEEK_END)
```

→ Apuntará al byte número 15.,

```
m.seek(-5, io.SEEK_SET)
```

→ Provocará un error de runtime (file pointer negativo).,

```
m.seek(5, io.SEEK_SET)
```

→ Apuntará al byte número 5.

Pregunta **12**
Correcta
Puntúa como 1

Una *Cola de Prioridad* es una cola en la cual los elementos se insertan en algún orden, pero tal que cuando se pide retirar un elemento se obtiene siempre *el menor* de los valores almacenados en la cola ¿Cuál de las siguientes estrategias podría ser una forma básica de implementar una *Cola de Prioridad* en las condiciones aquí expresadas?

Seleccione una:

a. Soporte: una variable de tipo *list* en Python. Inserción: *ordenada de menor a mayor* (mantener el arreglo siempre ordenado de menor a mayor: para insertar un valor *x*, recorrer el arreglo y al encontrar el primer mayor a *x* detener el ciclo y añadir allí a *x* con un corte de índices. Eliminación: siempre el primer elemento.

¡Correcto! Aunque entienda que si se hace así, logrará el objetivo pero la inserción tendrá un tiempo de ejecución O(n) mientras que la eliminación será de tiempo constante (O(1)).

b. Soporte: una varable de tipo *list* en Python. Inserción: siempre al frente. Eliminación: siempre el primer elemento.

c. Soporte: una variable de tipo *list* en Python. Inserción: *ordenada de mayor a menor* (mantener el arreglo siempre ordenado de mayor a menor: para insertar un nuevo valor *x*, recorrer el arreglo y al encontrar el primer menor, a *x* detener el ciclo y añadir allí a *x* con un corte de índices. Eliminación: siempre el primer elemento.

d. Soporte: una variable de tipo *list* en Python. Inserción: siempre al final. Eliminación: siempre el último elemento.

¡Correcto!

La respuesta correcta es:
Soporte: una variable de tipo *list* en Python. Inserción: *ordenada de menor a mayor* (mantener el arreglo siempre ordenado de menor a mayor: para insertar un valor *x*, recorrer el arreglo y al encontrar el primer mayor a *x* detener el ciclo y añadir allí a *x* con un corte de índices. Eliminación: siempre el primer elemento.

Pregunta **13**

Correcta

Puntúa como 1

¿Qué elementos son necesarios para que una *función recursiva* se considere *bien planteada*?

Seleccione una:

a. La función debe tener al menos una invocación a si misma en su bloque de acciones, y después de esas invocaciones debe tener una o más condiciones de control que permitan interrumpir el proceso recursivo si se ha llegado a alcanzar alguna de las situaciones triviales o base del problema.

b. La función deben tener al menos una invocación a si misma en su bloque de acciones.

c. La función debe tener al menos un ciclo en su bloque de acciones, y ese ciclo debe estar planteado de forma tal que nunca entre en un lazo infinito.

d. La función debe tener al menos una invocación a si misma en su bloque de acciones, y antes de esas invocaciones debe tener una o más condiciones de control que permitan interrumpir el proceso recursivo si se ha llegado a alcanzar alguna de las situaciones triviales o base del problema. ✔ ¡Correcto!

¡Correcto!

La respuesta correcta es:
La función debe tener al menos una invocación a si misma en su bloque de acciones, y antes de esas invocaciones debe tener una o más condiciones de control que permitan interrumpir el proceso recursivo si se ha llegado a alcanzar alguna de las situaciones triviales o base del problema.

Pregunta **14**

Correcta

Puntúa como 1

¿Cuáles de las siguientes son CIERTAS en relación al sistema de coordenadas de pantalla?

Seleccione una o más de una:

- [ ] a. El origen del sistema de coordenadas se encuentra en el punto inferior izquierdo de la pantalla o de la ventana que se esté usando.
- [ ] b. Las filas de pantalla o de ventana con número de orden más alto, se encuentran más cerca del borde superior que las filas con número de orden más bajo.
- [x] c. El origen del sistema de coordenadas se encuentra en el punto superior izquierdo de la pantalla o de la ventana que se esté usando. ¡Correcto!
- [x] d. Las filas de pantalla o de ventana con número de orden más bajo, se encuentran más cerca del borde superior que las filas con número de orden más alto. ¡Correcto!

¡Correcto!

Las respuestas correctas son:
El origen del sistema de coordenadas se encuentra en el punto superior izquierdo de la pantalla o de la ventana que se esté usando.,

Las filas de pantalla o de ventana con número de orden más bajo, se encuentran más cerca del borde superior que las filas con número de orden más alto.

Pregunta **15**

Incorrecta

Puntúa como 1

¿Cuál de las siguientes funciones creará una tupla conteniendo exclusivamente los números pares divisibles por 3 y por 7 al mismo tiempo, del intervalo [0, n]?

Seleccione una:

a.
```
def prueba(n):
    r = ()
    for i in range(0, n+1, 21):
        r += i,
    return r
```

✖ Incorrecto... así planteada, la tupla que se crea contiene a los múltiplos de 21 en [1, n]... que efectivamente son divisibles por 3 y por 7... pero muchos de ellos no son pares... de hecho, el propio 21 no es par...

b.
```
def prueba(n):
    r = ()
    for i in range(2, 3*(n+1), 7):
        r += i,
    return r
```

c.
```
def prueba(n):
    r = ()
    for i in range(0, n+1, 42):
        r += i,
    return r
```

d.
```
def prueba(n):
    r = ()
```

```
    for i in range(1, n+1, 21):
        r += i,
    return r
```

Revise el análisis general realizado en la Ficha 10, página 238 y siguientes, y ajústelo al caso de esta pregunta.

La respuesta correcta es:

```
def prueba(n):
    r = ()
    for i in range(0, n+1, 42):
        r += i,
    return r
```

Pregunta **16**

Incorrecta

Puntúa como 1

Analice el siguiente script básico en Python:

```
c1 = complex(3, 2)
c2 = complex('1 + 5j')
print('c1:', c1)
print('c2:', c2)
```

¿Cuál es el efecto de ejecutar el script anterior?

Seleccione una:

a. Ejecuta sin problemas, y muestra en consola de salida los complejos *c1: (3+2j)* y *c2: (1+5j)*.

b. Ejecuta sin problemas, y muestra en consola de salida los complejos *c1: (5j)* y *c2: (6j)*.

c. Al ejecutarse se interrumpirá en forma abrupta, lanzando un mensaje de cadena mal formada en la asignacion *c2 = complex('1 + 5j')* (la cadena contiene espacios en blanco a los lados del signo '+').

d. Al ejecutarse se interrumpirá en forma abrupta, lanzando un mensaje de operación imposible en la primera asignación *c1 = complex(3, 2)* (falta el factor *j* acompañando al 2). ✖

Incorrecto... si se usa la función complex() para crear un complejo, pasándole los números que lo formarán, el factor j no debe enviarse: Python lo asume y lo agrega automáticamente.

Revise la Ficha 6, página 130.

La respuesta correcta es:
Al ejecutarse se interrumpirá en forma abrupta, lanzando un mensaje de cadena mal formada en la asignacion *c2 = complex('1 + 5j')* (la cadena contiene espacios en blanco a los lados del signo '+').

Pregunta **17**

Correcta

Puntúa como 1

De las que siguen, seleccione la afirmación **incorrecta** respecto a las bajas de registros en archivos binarios en Python.

Seleccione una:

a. No disponemos de una instrucción en Python que permita eliminar un registro del archivo, en forma similar a lo que hace la instrucción **del** para eliminar un componente de un arreglo representado en una variable de tipo list.

b. Se suele aplicar un esquema combinado bajas lógicas y bajas físicas. Las bajas lógicas realzarlas en algún momento no crítico del uso del archivo, por ejemplo, cuando está cerrada la organización, sin atención al público. Las bajas físicas realizarlas en tiempo real, cuando la organización está atendiendo al público, para un mejor rendimiento del tiempo. ✔

¡Correcto!
Efectivamente, esta respuesta está MAL, ya que el esquema se aplica exactamente al revés.

c. Para hacer bajas físicas, se puede utilizar un archivo temporal, inicialmente vacío, en el cual se copien todos los registros del archivo original, menos el o los que se deseen borrar. Luego se destruye el archivo original y se cambia el nombre del archivo temporal por el nombre que tenía el archivo original.

d. Para una baja lógica, el registro que se quiere dar de baja no se elimina físicamente, sino que se lo marca usando un campo especialmente definido a modo de bandera.

¡Correcto!

La respuesta correcta es:
Se suele aplicar un esquema combinado bajas lógicas y bajas físicas. Las bajas lógicas realzarlas en algún momento no crítico del uso del archivo, por ejemplo, cuando está cerrada la organización, sin atención al público. Las bajas físicas realizarlas en tiempo real, cuando la organización está atendiendo al público, para un mejor rendimiento del tiempo.

Pregunta **18**

Incorrecta

Puntúa como 1

Para problema general nombrado en la columna de la izquierda, seleccione la estrategia de planteo de algoritmos que se sabe haya resultado más útil para resolver ese problema, o bien la que sea que haya podido aplicarse para resolverlo aún sin llegar a una solución eficiente (considere a cada problema en su situación más general, y no casos particulares de cada uno):

Generación de gráficos fractales.	Backtracking ✖
Problema de las Ocho Reinas	Divide y vencerás ✖
Problema del árbol de expansión mínimo de un grafo.	Fuerza Bruta [O(n!)] / Programación Dinámica [O(n^2 * 2^n)] ✖
Problema del Viajante.	Programación dinámica ✖
Ordenamiento rápido (Quicksort).	Algoritmo ávido ✖
Problema de la alineación de secuencias.	Recursión ✖

Revise la Ficha 28, páginas 574 a 575...

La respuesta correcta es: Generación de gráficos fractales. → Recursión, Problema de las Ocho Reinas → Backtracking, Problema del árbol de expansión mínimo de un grafo. → Algoritmo ávido, Problema del Viajante. → Fuerza Bruta [O(n!)] / Programación Dinámica [O(n^2 * 2^n)], Ordenamiento rápido (Quicksort). → Divide y vencerás, Problema de la alineación de secuencias. → Programación dinámica

Pregunta **19**

Correcta

Puntúa como 1

El siguiente programa crea y carga un arreglo *temp* con *n* valores de temperaturas medidas en diferentes momentos y luego procesa ese arreglo mediante la función *amplitud()* ¿Qué hace exactamente esa función?

```
__author__ = 'Cátedra de AED'


def read(temp):
    n = len(temp)
    for i in range(n):
        temp[i] = int(input('Temperatura[' + str(i) + ']: '))


def amplitud(temp):
    n = len(temp)
    my = mn = temp[0]
    for i in range(1, n):
        if temp[i] > my:
            my = temp[i]
        elif temp[i] < mn:
            mn = temp[i]

    return my - mn


def test():
    n = int(input('Cantidad de temperaturas a cargar: '))
    temp = n * [0.0]
    read(temp)

    d = amplitud(temp)

    print('Amplitud térmica:', d)


if __name__ == '__main__':
    test()
```

Seleccione una:

a. Calcula y retorna la diferencia entre la mayor y la menor temperatura del arreglo *temp*. ✔ ¡Correcto!

b. Calcula y retorna la menor temperatura del arreglo *temp*.

c. Calcula y retorna el promedio entre la mayor y la menor temperatura del arreglo *temp*.

d. Calcula y retorna la la mayor temperatura del arreglo *temp*.

¡Correcto!

La respuesta correcta es:
Calcula y retorna la diferencia entre la mayor y la menor temperatura del arreglo *temp*.

Pregunta **20**

Correcta

Puntúa como 1

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, 'Cantidad de socios:', vc[i])
```

Seleccione una:

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

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

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 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].

¡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 **21**

Correcta

Puntúa como 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. De acuerdo a cada sistema operativo, ese momento inicial del tiempo puede ser establecido en fechas y horas diferentes. ¡Ok!
- b. 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.
- c. 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.
- d. 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.

¡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.

Pregunta **22**

Correcta

Puntúa como 1

¿Qué significa decir que un algoritmo dado tiene un tiempo de ejecución **O(log(n))**?

Seleccione una:

a. El proceso normalmente consiste en dos ciclos (uno dentro del otro) de aproximadamente n iteraciones cada uno, de forma que las operaciones críticas se aplican un número cuadrático de veces.

b. El tiempo de ejecucíón es constante, sin importar la cantidad de datos.

c. A medida que aumenta el número de datos, aumenta el tiempo pero en forma muy suave: el conjunto de datos se divide en dos. se procesa una de las mitades, se desecha la otra y se repite el proceso hasta que no pueda volver a dividirse la mitad que haya quedado. ¡Correcto!

d. El tiempo de ejecución es lineal: si aumenta el número de datos, aumenta el tiempo en la misma proporción.

¡Correcto!

La respuesta correcta es:
A medida que aumenta el número de datos, aumenta el tiempo pero en forma muy suave: el conjunto de datos se divide en dos. se procesa una de las mitades, se desecha la otra y se repite el proceso hasta que no pueda volver a dividirse la mitad que haya quedado.

Pregunta **23**

Correcta

Puntúa como 1

Suponga la función *test()* (que se muestra más abajo) que toma como parámetro una matriz *mat* ya creada con valores de tipo *int*. ¿Qué hace concretamente función?

```
def test(mat):
    n = len(mat)
    m = len(mat[0])

    if n != m:
        return False

    for i in range(n):
        if mat[i][i] != 0:
            return False

    return True
```

Seleccione una:

a. Comprueba si la matriz es cuadrada. En caso de serlo, retorna *True* si la diagonal principal de esa matriz contiene todos sus componentes valiendo distinto de *0(cero)*. Retorna *False* si la matriz no es cuadrada o si alguno de los casilleros de la diagonal contiene el valor cero.

b. Comprueba si la matriz es cuadrada. Retorna *True* si lo es, o retorna *False* en caso contrario.

c. Comprueba si la matriz es cuadrada. En caso de serlo, retorna *True* si la diagonal principal de esa matriz contiene todos sus componentes valiendo *0(cero)*. Retorna *False* si la matriz no es cuadrada o si alguno de los casilleros de la diagonal contiene un valor diferente a cero. ¡Correcto!

d. Comprueba si la matriz es cuadrada. En caso de serlo, retorna *True* si todos los casilleros de esa matriz valen *0(cero)*. Retorna *False* si la matriz no es cuadrada o si alguno de los casilleros de la matriz contiene un valor diferente a cero.

¡Correcto!

La respuesta correcta es:
Comprueba si la matriz es cuadrada. En caso de serlo, retorna *True* si la diagonal principal de esa matriz contiene todos sus componentes valiendo *0(cero)*. Retorna *False* si la matriz no es cuadrada o si alguno de los casilleros de la diagonal contiene un valor diferente a cero.

Pregunta **24**

Correcta

Puntúa como 1

En la columna de la izquierda se muestra el código fuente en Python de diversos procesos sencillos. Para cada uno de ellos, seleccione de la lista de la derecha la expresión en notación *Big O* que mejor describa el tiempo de ejecución de cada proceso para el peor caso. (Aclaración: una expresión como *n^2* debe entenderse como "*n al cuadrado*" o n^2).

```
ac = 0
for i in range(10):
    ac += i
print(ac)
```

O(1) ✔

```
n = int(input('N: '))
ac = 0
for i in range(n):
    for j in range(i+1, n):
        ac += i*j

for i in range(n):
    for j in range(n):
        for k in range(n):
            ac += (i+j+k)
print(ac)
```

O(n^3) ✔

```
n = int(input('N: '))
ac = 0
for i in range(n):
    for j in range(i+1, n):
        ac += i*j
print(ac)
```

O(n^2) ✔

```
n = int(input('N: '))
ac = 0
for i in range(n):
    ac += i
print(ac)
```

O(n) ✔

¡Correcto!

La respuesta correcta es:

```
ac = 0
for i in range(10):
    ac += i
print(ac)
```

→ O(1),

```
n = int(input('N: '))
ac = 0
for i in range(n):
    for j in range(i+1, n):
        ac += i*j

for i in range(n):
    for j in range(n):
        for k in range(n):
            ac += (i+j+k)
print(ac)
```

→ O(n^3),

```
n = int(input('N: '))
ac = 0
for i in range(n):
    for j in range(i+1, n):
        ac += i*j
print(ac)
```

→ O(n^2),

```
n = int(input('N: '))
ac = 0
for i in range(n):
    ac += i
print(ac)
```

→ O(n)

Pregunta **25**

Correcta

Puntúa como 1

En general, una ***expresión*** es una fórmula en la cual se usan *operadores* (como suma, resta, producto, compración, etc.) sobre diversas variables y constantes (que reciben el nombre de *operandos* de la expresión). Son ejemplos válidos los siguientes: 3 * a + 2, b / c - 4, (7 - r) / (4 + a), a > b, x + 2 >= 10.

¿Es correcta la siguiente definición?

"Una ***expresión aritmética*** es una expresión en la cual el resultado final es un número"

Seleccione una:

- (•) Verdadero ✔
- ( ) Falso

¡Correcto!

La respuesta correcta es 'Verdadero'

◄ Examen Final - Registro de Calificación Final (Libres, Regulares y Promocionados)

Ir a...

Enunciado Examen Final Práctico [Regulares] ►