

**Comenzado el** jueves, 19 de abril de 2018, 00:48

**Estado** Finalizado

**Finalizado en** sábado, 21 de abril de 2018, 11:57

**Tiempo empleado** 2 días 11 horas

**Puntos** 23/23

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 2 sobre 2

¿Qué hace el siguiente script en Python?

```
cad = 'Python'  
print(cad[1], cad[4])  
print(cad[1] + cad[4])
```

Seleccione una:

a.

Produce un error de intérprete, ya que no se puede acceder a los elementos individuales de una cadena mediante índices encerrados entre corchetes.

b.

Accede al carácter en la posición 1 ('y') y al carácter en la posición 4 ('o') de la cadena. El primer print() los muestra *por separado* (y o) pero el segundo los muestra *concatenados* (yo). ✓

¡Ok!

c.

Accede al carácter en la posición 1 ('y') y al carácter en la posición 4 ('o') de la cadena. Ambos print() los muestran *por separado* (y o).

d.

Accede al primer carácter de la cadena ('P') y al cuarto carácter de la cadena ('h'). El primer print() los muestra *por separado* (P h) pero el segundo los muestra *concatenados* (Ph).

¡Correcto!

La respuesta correcta es:

Accede al carácter en la posición 1 ('y') y al carácter en la posición 4 ('o') de la cadena. El primer print() los muestra *por separado* (y o) pero el segundo los muestra *concatenados* (yo).

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

¿Cuál de las siguientes asignaciones de cadenas de caracteres en una variable provocará un error de intérprete en Python?

Seleccione una:

a.

cad = 'Ray "Sugar" Leonard'

b.

cad = "Ray \'Sugar\' Leonard"

c.

cad = \'Ray "Sugar" Leonard\' ✓

¡Ok! Efectivamente... el uso del carácter de escape \' permite incluir una comilla simple dentro de una cadena, pero suponiendo que la misma ya esté delimitada con comillas o apóstrofos...

d.

cad = "Ray 'Sugar' Leonard"

¡Correcto!

La respuesta correcta es:

cad = \'Ray "Sugar" Leonard\'

**Pregunta 3**

Correcta

Puntúa 2 sobre 2

¿Hay algún problema en Python con la siguiente secuencia de instrucciones de asignación de cadenas de caracteres?

```
res = 'toro'  
print('Valor: ', res)  
  
res[0] = 'c'  
print('Valor: ', res)
```

Seleccione una:

a.

El script mostrado lanza un error de ejecución en la asignación `res[0] = 'c'`: las cadenas de caracteres son inmutables, y por lo tanto no se puede cambiar un carácter mediante índices entre corchetes



¡Ok! Efectivamente...

b.

El script mostrado no lanza ningún error, pero la variable `res` termina valiendo la cadena 'toro' inicialmente asignada. La asignación del carácter 'c' será ignorada, debido a que las cadenas de caracteres son inmutables.

c.

No. No hay ningún problema.

d.

El script mostrado lanza un error de intérprete en la asignación `res = 'toro'`: las cadenas de caracteres no pueden ser asignadas con el operador `=`.

¡Correcto!

La respuesta correcta es:

El script mostrado lanza un error de ejecución en la asignación `res[0] = 'c'`: las cadenas de caracteres son inmutables, y por lo tanto no se puede cambiar un carácter mediante índices entre corchetes

**Pregunta 4**

Correcta

Puntúa 2 sobre 2

Considere la siguiente instrucción de visualización en pantalla para Python

3:

```
print('Hola\nMundo...\n\t...otra vez')
```

¿Cuál de las siguientes salidas por consola estándar será la que se produzca con la instrucción anterior?

Seleccione una:

 a.

Hola

Mundo...

...otra vez 

¡Ok! Efectivamente... cada \n produce un salto de línea (por eso habrá tres líneas de salida) y el carácter \t produce un espaciado horizontal en la tercera antes de la cadena '...otra vez'.

 b.

Hola Mundo...

...otra vez

 c.

Hola Mundo...

...otra vez

 d.

Hola Mundo... ...otra vez

¡Correcto!

La respuesta correcta es:

Hola

Mundo...

...otra vez

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes NO ES una recomendación de estilo de escritura de código fuente en Python de acuerdo a la *Guía PEP 8*?

Seleccione una:

a.

Colocar los comentarios de texto en líneas específicas para esos comentarios, de ser posible.

b.

Usar líneas en blanco para separar largos bloques de código fuente.

c.

Mantener consistencia en el uso de comillas dobles o simples para manejar cadenas: si se comenzó con una de ellas, mantenerse con ella a lo largo del programa.

d.

No colocar espacios alrededor de los operadores y los operandos de una expresión: escribir la expresión de corrido, sin espacios separadores. ✓

¡OK! Justamente, la Guía PEP 8 sugiere lo contrario: use espaciado y aplique sentido común.

¡Correcto!

La respuesta correcta es:

No colocar espacios alrededor de los operadores y los operandos de una expresión: escribir la expresión de corrido, sin espacios separadores.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Cuál es el nombre del creador del lenguaje de programación *Python*?

Seleccione una:



a.



James Gosling



b.



Guido van Rossum ✓

¡Ok!



c.



Ada Byron



d.



Niklaus Wirth

¡Correcto!

La respuesta correcta es:



Guido van Rossum

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes es el nombre informal alternativo con el cual se conoce al documento *PEP 20* de la documentación oficial de Python? (Claramente, esta es una pregunta para distenderse... nadie espera que se equivoquen aquí... sólo tómense un minuto para buscar la respuesta... y mientras, ríanse un poco con las opciones que les estamos sugiriendo...)

Seleccione una:

- a.  
El Zen de Python ✓  
¡Ok!
- b.  
El Kung Fu Python
- c.  
El Ninja Python
- d.  
El Python Saiyajin

¡Correcto!

La respuesta correcta es:

El Zen de Python

**Pregunta 8**

Correcta

Puntúa 2 sobre 2

¿Qué efecto provoca la siguiente secuencia de instrucciones en Python?

```
a = 5  
b = 3  
c = a  
a = b  
b = c
```

Seleccione una:

 a.

Quedan con los siguiente valores: a = 3 b = 5 c = 5.



¡Ok!

 b.

Quedan con los siguiente valores: a = 5 b = 3 c = 5

 c.

Las tres variables (a, b, c) quedan valiendo 3.

 d.

Las tres variables (a, b, c) quedan valiendo 5.

¡Correcto! Observe que el efecto general de esta secuencia, es que se intercambian los valores iniciales de **a** y **b** (y en este caso, no importa demasiado cuánto queda valiendo **c**). Y recuerde que en Python podría haber logrado el mismo efecto haciendo simplemente **a, b = b, a** (asignación de tuplas).

La respuesta correcta es:

Quedan con los siguiente valores: a = 3 b = 5 c = 5.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

Sabemos que **Ada Augusta Byron King (Condesa de Lovelace)** fue la hija del poeta Lord Byron y actuó como colaboradora de Charles Babbage en el diseño de la Analytical Engine, proponiendo elementos conceptuales para la programación de esa máquina que hoy se usan en los modernos lenguajes de programación (subrutinas y ciclos, por ejemplo). ¿Cómo se llama el lenguaje de programación moderno designado así en honor a ella?

Seleccione una:

- a.  
Lovelace
- b.  
Byron
- c.  
Ada ✓  
¡Ok!
- d.  
Augusta

¡Correcto!

La respuesta correcta es:

Ada

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

El Premio Turing (otorgado anualmente por la ACM - Association for Computing Machinery a quienes hayan realizado aportes trascendentales en el campo de las Ciencias de la Computación) está considerado como el equivalente al Premio Nobel de las Ciencias de la Computación. ¿Cuál de los siguientes famosos exponentes del mundo de la Ciencias de la Computación ganó oportunamente el Premio Turing por haber sido el creador de varios lenguajes de programación innovadores (entre ellos: *Pascal*, *Modula* y *Algol-W*)?

Seleccione una:

a.



Niklaus Wirth ✓

¡Ok! Un genio...

b.



Frances Elizabeth (Fran) Allen

c.



Charles Antony (Tony) Richard Hoare

d.



Adi Shamir

¡Correcto!

La respuesta correcta es:



Niklaus Wirth

**Pregunta 11**

Correcta

Puntúa 2 sobre 2

¿Hay algún problema en Python con la siguiente secuencia de instrucciones de asignación de cadenas de caracteres?

```
res = 'toro'  
print('Valor: ', res)  
  
res = 'coro'  
print('Valor: ', res)
```

Seleccione una:

a.

El script mostrado lanza un error de intérprete en la asignación `res = 'coro'`: las cadenas de caracteres son inmutables, y por lo tanto `res` no puede cambiar de valor.

b.

No. No hay ningún problema.



¡Ok! Efectivamente... la variable `res` comienza apuntando a la cadena '`'toro'`', y luego pasa a apuntar a la cadena '`'coro'`'. No hay ningún problema.

c.

El script mostrado lanza un error de intérprete en la asignación `res = 'toro'`: las cadenas de caracteres no pueden ser asignadas con el operador `=`.

d.

El script mostrado no lanza ningún error, pero la variable `res` termina valiendo la cadena '`'toro'`' inicialmente asignada. La asignación de la cadena '`'coro'`' será ignorada, debido a que las cadenas de caracteres son inmutables.

¡Correcto!

La respuesta correcta es:

No. No hay ningún problema.

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes tipos estructurados es un tipo de secuencia *mutable* en Python?

Seleccione una:

- a.  
Rango (range)
- b.  
Cadenas de caracteres (str)
- c.  
Tuplas (tuples)
- d.  
Lista (list)



¡Ok!

¡Correcto!

La respuesta correcta es:

Lista (list)

**Pregunta 13**

Correcta

Puntúa 2 sobre 2

¿Qué hace el siguiente script en Python?

```
a = int(input('a: '))
b = int(input('b: '))
c = int(input('c: '))
d = int(input('d: '))

m = min(max(a, b), max(c, d))
print('Resultado: ', m)
```

Seleccione una:

- a.

Calcula la *suma* entre los valores *a* y *b*, luego la *suma* entre los valores *c* y *d*, finalmente obtiene el *menor* entre ambas *sumas* parciales, y lo muestra.

- b.

Calcula el *mayor* entre los valores *a* y *b*, luego el *mayor* entre los valores *c* y *d*, finalmente obtiene el *menor* entre esos dos *mayores*, y lo muestra. ✓

¡Ok!

- c.

Calcula el *menor* entre los valores *a* y *b*, luego el *menor* entre los valores *c* y *d*, finalmente obtiene el *mayor* entre esos dos *menores*, y lo muestra.

- d.

Calcula el *menor* entre los valores *a* y *b*, luego el *mayor* entre los valores *c* y *d*, finalmente obtiene el *menor* entre los dos que obtuvo, y lo muestra.

¡Correcto!

La respuesta correcta es:

Calcula el *mayor* entre los valores *a* y *b*, luego el *mayor* entre los valores *c* y *d*, finalmente obtiene el *menor* entre esos dos *mayores*, y lo muestra.

**Pregunta 14**

Correcta

Puntúa 3 sobre 3

Considere la secuencia de instrucciones Python que se muestra a continuación:

```
a = 10  
b = 20  
c = 30
```

¿Cuáles de las siguientes instrucciones o secuencias de instrucciones **NO PROVOCAN** un cambio en el valor final de la variable **b**? (Observación: más de una respuesta puede ser correcta. Marque **todas** las que considere aplicables)

Seleccione una o más de una:



a.

 $a, b, c = c, b, a$  ✓

Correcto. En este caso, la única variable que no cambia de valor es justamente **b**...



b.

 $d = b, a, c$  $a, b, c = d$ 

c.

 $a, b = a, b$  ✓

Ok. Tanto **a** como **b** quedan con sus valores iniciales.



d.

 $d = c, a, b$  $a, b, c = d$ 

¡Correcto!

Las respuestas correctas son:

 $a, b = a, b,$  $a, b, c = c, b, a$

**Comenzado el** martes, 24 de abril de 2018, 00:10

**Estado** Finalizado

**Finalizado en** martes, 24 de abril de 2018, 19:56

**Tiempo empleado** 19 horas 46 minutos

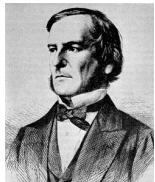
**Puntos** 20/20

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1



¿Cuáles fueron los aportes que realizaron *George Boole* y *Augustus De Morgan* en el campo matemático del tratamiento de las relaciones lógicas?

Seleccione una:

a.

*Boole* sentó las bases del Álgebra de Boole, y *De Morgan* demostró que el Álgebra de Boole es válida.

b.

*Boole* sentó las bases del Álgebra de Boole, y *De Morgan* planteó y demostró las leyes de De Morgan para negar conjunciones y disyunciones. ✓

¡Ok!

c.

*De Morgan* sentó las bases del Álgebra de De Morgan, y *Boole* planteó y demostró las leyes de Boole para negar conjunciones y disyunciones.

d.

*Boole* sentó las bases de la aritmética en sistema binario, y *De Morgan* usó el sistema binario para diseñar la primera computadora digital de la historia.

¡Correcto!

La respuesta correcta es:

*Boole* sentó las bases del Álgebra de Boole, y *De Morgan* planteó y demostró las leyes de De Morgan para negar conjunciones y disyunciones.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

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

¿Es correcta la siguiente definición?

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

Seleccione una:

- Verdadero
- Falso ✓

¡Correcto!

La respuesta correcta es 'Falso'

**Pregunta 3**

Correcta

Puntúa 2 sobre 2

¿Cuál de las siguientes expresiones lógicas es verdadera **si y sólo si** el valor de la variable **x** es 1, 2, 3, o 4?

Seleccione una:

- a.  
 $x == 1 \text{ or } x == 2 \text{ or } x == 3 \text{ or } x == 4$  ✓  
¡Ok!
- b.  
 $x != 1 \text{ and } x != 2 \text{ and } x != 3 \text{ and } x != 4$
- c.  
 $x == 1 \text{ and } x == 2 \text{ and } x == 3 \text{ and } x == 4$
- d.  
 $x < 0 \text{ and } x > 5$

¡Correcto!

La respuesta correcta es:

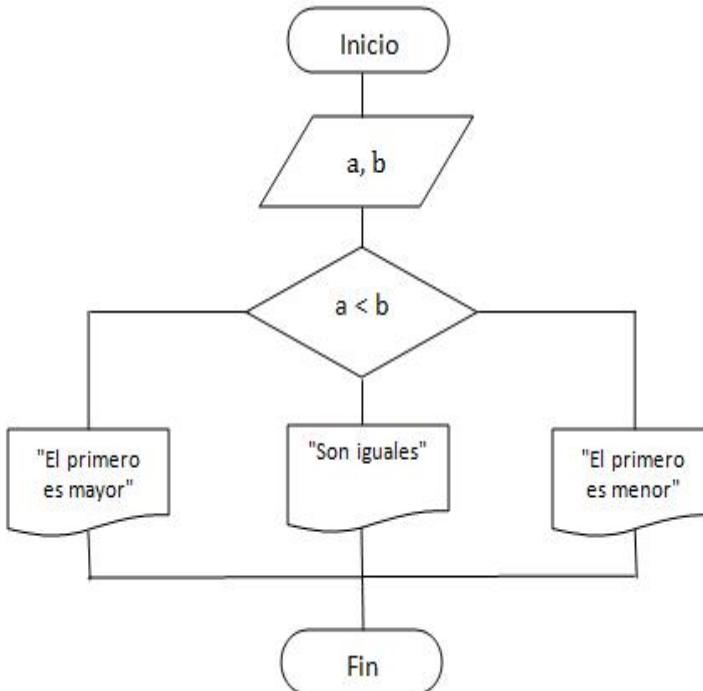
$x == 1 \text{ or } x == 2 \text{ or } x == 3 \text{ or } x == 4$

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Suponga que se desea desarrollar un programa que cargue dos números y muestre un mensaje indicando si el primero es menor, igual o mayor al segundo. ¿Está bien planteado el siguiente diagrama de flujo?



Seleccione una:

- a.  
Sí. El diagrama está correctamente planteado.
- b.  
Está mal planteado: en la condición, se debía preguntar si  $a \geq b$ .
- c. Está mal planteado: la condición está mal dibujada, ya que una condición no puede tener tres salidas o ramas. ✓ ¡Ok!
- d.  
Está mal planteado: el símbolo usado para la carga de datos, debió ser un rectángulo y no un paralelogramo.

¡Correcto!

La respuesta correcta es: Está mal planteado: la condición está mal dibujada, ya que una condición no puede tener tres salidas o ramas.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

Sean las siguientes variables:

 $a, b, c = 3, 10, 2$ 

¿Cuáles de las siguientes expresiones lógicas obtendrán un valor final **True** usando las variables y valores aquí indicados? (Observación: puede haber VARIAS respuestas correctas... marque TODAS las que considere válidas)

Seleccione una o más de una:

 a. $a == c \text{ and } b == 10 \text{ and } c != 8$  b. $a == 3 \text{ or } b > 100 \text{ or } c != 2$  ✓

¡Ok!

 c. $a != b \text{ and } b != 0 \text{ and } c >= 1$  ✓

¡Ok!

 d. $a >= b \text{ or } b == 2*c \text{ or } (c == 2 \text{ and } a == 4)$ 

¡Correcto!

Las respuestas correctas son:

 $a != b \text{ and } b != 0 \text{ and } c >= 1,$  $a == 3 \text{ or } b > 100 \text{ or } c != 2$

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Por qué motivo debe *indentarse* (*encolumnarse hacia la derecha*) correctamente cada rama de una instrucción condicional en Python?

Seleccione una:

a.

No es cierto que se deba indentar cada rama. La indentación sólo se sugiere por razones de claridad.

b.

Para que Python pueda reconocer qué instrucciones pertenecen a cada rama. Pero aún así, la indentación sólo es obligatoria en las ramas que tengan una y sólo una instrucción.

c.

Para que Python pueda reconocer qué instrucciones pertenecen a cada rama. ✓

¡Ok!

d.

Para que Python pueda reconocer qué instrucciones pertenecen a cada rama. Pero aún así, la indentación sólo es obligatoria en las ramas que tengan más de una instrucción.

¡Correcto!

La respuesta correcta es:

Para que Python pueda reconocer qué instrucciones pertenecen a cada rama.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Esta pregunta está orientada a la aplicación de las Leyes de Morgan para negar una expresión lógica formada por conjunciones y disyunciones. Sea la siguiente expresión, sumiendo que las variables que se indican están correctamente inicializadas y con valores numéricos:

```
r = not(a < c or b == 5*d + 2 or not(e >= a and f != d) or d
< c // 4)
```

¿Cuál de las siguientes expresiones lógicas es equivalente a la expresión anterior? (o sea, ¿cuál de ellas tiene la misma tabla de verdad?) Aplique las Leyes de De Morgan con paciencia y cuidado. Recomendamos asignar valores a las variables, y probar con cada expresión que logre obtener para comparar los resultados que obtenga.

Seleccione una:

a.

```
r = a >= c and b != 5*d + 2 and e < a and f == d and d
>= c // 4
```

b.

```
r = a >= c and b != 5*d + 2 and e < a or f == d and d >=
c // 4
```

c.

```
r = a >= c and b != 5*d + 2 and e >= a and f != d and d
>= c // 4 ✓
```

¡Ok!

d.

```
r = a >= c or b != 5*d + 2 or e >= a or f != d or d >=
c // 4
```

¡Correcto!

La respuesta correcta es:

```
r = a >= c and b != 5*d + 2 and e >= a and f != d and d >=
c // 4
```

**Pregunta 8**

Correcta

Puntúa 3 sobre 3

¿Qué tiene de malo el siguiente script en Python?

```
x1 = int(input('Primer valor: '))
x2 = int(input('Segundo valor: '))

if x1 = x2:
    print('Son iguales')
else:
    print('No son iguales')
```

Seleccione una:

 a.

Al ejecutar, la condición sale siempre por falso.

 b.

Al ejecutar, la condición sale siempre por verdadero.

 c.

Los mensajes que muestra en ambas ramas están al revés.

 d.No compila: usa el operador = para comparar, cuando debió usar el == (doble igual). 

¡Ok! efectivamente, en Python esto provoca un error de intérprete.

A diferencia de otros lenguajes, en Python el operador de asignación no puede usarse en una expresión lógica, incluso si las variables fuesen booleanas.

¡Correcto!

La respuesta correcta es:

No compila: usa el operador = para comparar, cuando debió usar el == (doble igual).

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Cuál es el efecto del conector and ("y lógico") en una condición?

Seleccione una:

a.

La condición es verdadera si todas las proposiciones son verdaderas.



¡Ok!

b.

La condición es verdadera si alguna de las proposiciones es verdadera.

c.

La condición es verdadera si todas las proposiciones son falsas.

d.

La condición es verdadera si una y sólo una de las proposiciones es verdadera.

¡Correcto!

La respuesta correcta es:

La condición es verdadera si todas las proposiciones son verdaderas.

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

¿Cuál es el efecto del operador or ("o lógico") en una condición?

Seleccione una:

a.

La condición es verdadera si al menos una de las proposiciones es verdadera. ✓

¡Ok!

b.

La condición es verdadera si y sólo si todas las proposiciones son verdaderas.

c.

La condición es verdadera si y sólo si una única proposición es verdadera. Si más de una es verdadera, la salida es False.

d.

La condición es verdadera si todas las proposiciones son falsas.

¡Correcto!

La respuesta correcta es:

La condición es verdadera si al menos una de las proposiciones es verdadera.

**Pregunta 11**

Correcta

Puntúa 2 sobre 2

El siguiente script en Python, pretende dejar en la variable *may* el mayor de los valores contenidos en las variables *n1* y *n2*. De acuerdo a esto... ¿Hay algún problema con el script mostrado?

```
n1 = int(input('Primer valor: '))
n2 = int(input('Segundo valor: '))

if n1 > n2:
    may = n1
else:
    may = n1

print('Mayor: ', may)
```

Seleccione una:

a.

Está mal planteada la rama falsa: está asignando *n1* cuando debería asignar *n2*. ✓

¡Ok!

b.

Está mal escrita la rama falsa: debió usar *elif* en lugar de *else*.

c.

Está mal planteada la rama verdadera: está asignando *n1* cuando debería asignar *n2*.

d.

No hay ningún problema: calcula y muestra correctamente el mayor, en todos los casos.

¡Correcto!

La respuesta correcta es:

Está mal planteada la rama falsa: está asignando *n1* cuando debería asignar *n2*.

**Pregunta 12**

Correcta

Puntúa 2 sobre 2

Suponga que se le pide desarrollar un programa que sea capaz de *elegir aleatoriamente una carta cualquiera de una (y sólo una) baraja española*. ¿Hay algún inconveniente con el programa que les mostramos aquí, o en líneas generales cumple con el requerimiento?

```
__author__ = 'Cátedra de AED'

import random

# Título principal...
print('Selección aleatoria de una carta de la baraja
española...')

# Selección del número de la carta...
n = random.randint(1, 12)

# Selección del palo de la carta...
palos = 'Espada', 'Basto', 'Oro', 'Copa'
p = random.choice(palos)

# Visualización de resultados...
print('La carta seleccionada es:')
print('Palo:', p, '- Valor:', n)
```

Seleccione una:

- a.  
El programa cumple con el requerimiento sin ningún inconveniente, seleccionando siempre cartas diferentes en ejecuciones diferentes o repitiendo el mismo esquema en el mismo programa.
- b.  
El programa cumple con el requerimiento (aunque un inconveniente es que podría repetir la misma carta en dos ejecuciones diferentes o repitiendo el mismo esquema en el mismo programa). ✓  
¡Ok! Efectivamente, selecciona bien una carta cualquiera, pero si se ejecuta nuevamente (o se repite el mismo esquema en el mismo programa) todas las cartas vuelven a estar disponibles y podría volver a seleccionar la misma.
- c.  
El programa no cumple correctamente con el requerimiento: a veces selecciona incorrectamente el número o valor de la carta.
- d.  
El programa no cumple correctamente con el requerimiento: a veces selecciona incorrectamente el palo de la carta.

¡Correcto!

La respuesta correcta es:

El programa cumple con el requerimiento (aunque un inconveniente es que podría repetir la misma carta en dos ejecuciones diferentes o repitiendo el mismo esquema en el mismo programa).

**Pregunta 13**

Correcta

Puntúa 2 sobre 2

¿Qué hace el script que se muestra en el siguiente esquema?

```
__author__ = 'Cátedra de AED'

import random

print('Ejemplo de uso de random.random()...')
f = random.random()
i = int(f * 10)
print('El valor generado es:', i)
```

Seleccione una:

- a.  
Genera aleatoriamente un número en coma flotante en el intervalo [0, 1), y lo asigna en la variable *i*.
- b.  
Genera aleatoriamente un número en coma flotante en el intervalo [1, 10), y lo asigna en la variable *i*.
- c.  
Genera aleatoriamente un número entero en el intervalo [0, 9], y lo asigna en la variable *i*. ✓  
¡Ok! Efectivamente... lo primero que hace es invocar a `random.random()` y obtiene con eso un número con decimales entre 0 y 1... pero luego lo multiplica por 10 (con lo cual la coma se corre a la derecha exactamente un lugar, eliminando el cero y tomando el primer decimal para la parte entera) y finalmente trunca los decimales con la función `int()`. El resultado es un número entero de un sólo dígito, en [0, 9].
- d.  
Genera aleatoriamente un número entero en el intervalo [1, 10], y lo asigna en la variable *i*.

¡Correcto!

La respuesta correcta es:

Genera aleatoriamente un número entero en el intervalo [0, 9], y lo asigna en la variable *i*.



**Comenzado el** lunes, 7 de mayo de 2018, 15:37

**Estado** Finalizado

**Finalizado en** lunes, 7 de mayo de 2018, 15:52

**Tiempo empleado** 14 minutos 47 segundos

**Puntos** 15/15

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿En qué famoso museo de la ciudad de Londres (Inglaterra) se ha construido y se expone al público una *Analytical Engine*, la máquina diseñada por *Charles Babbage* a principios del siglo XIX y que es la precursora de las modernas computadoras? [El museo construyó la máquina usando materiales que estaban disponibles en la época en que Babbage la diseñó...]

Seleccione una:

- a.  
Museo de Guerra
- b.  
Museo de Ciencias ✓  
¡Ok!
- c.  
Museo Británico
- d.  
Museo de Cera

¡Correcto!

La respuesta correcta es:

Museo de Ciencias

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuántas condiciones **como mínimo** necesita en un programa en Python para encontrar el menor valor de entre tres variables (sin usar la función `min()` de la librería estándar)?

Seleccione una:

- a.  
Una condición.
- b.  
Dos condiciones. ✓  
¡Ok! Puede plantearlo de varias formas, pero le quedarán dos condiciones al final...
- c.  
Cuatro condiciones.
- d.  
Tres condiciones.

¡Correcto!

La respuesta correcta es:

Dos condiciones.

**Pregunta 3**

Correcta

Puntúa 2 sobre 2

¿Qué hace siguiente script en Python?

```
a = int(input('Ingrese limite izquierdo: '))
b = int(input('Ingrese limite derecho: '))
if a > b:
    a, b = b, a

x = int(input('Ingrese el valor a chequear: '))
if a <= x <= b:
    print('Ok...')
else:
    print('Revise...')
```

Seleccione una:

 a.

Determina si el valor  $x$  es mayor que los valores  $a$  y  $b$ . Muestra "Ok..." si es cierto, o "Revise..." si no es cierto.

 b.

Determina si  $x$  es igual al promedio de los valores  $a$  y  $b$ . Muestra el mensaje "Ok..." si es cierto, o el mensaje "Revise..." si no es así.

 c.

Determina si el valor  $x$  está contenido en el intervalo  $[a, b]$ . Muestra el mensaje "Ok..." si es así, o muestra "Revise..." si no es cierto. ✓

¡Ok!

 d.

Ordena los valores  $a$ ,  $b$ ,  $x$  y muestra el valor "Ok..." al final si quedaron ordenados de menor a mayor, o bien muestra "Revise..." si quedaron de mayor a menor.

¡Correcto!

La respuesta correcta es:

Determina si el valor  $x$  está contenido en el intervalo  $[a, b]$ . Muestra el mensaje "Ok..." si es así, o muestra "Revise..." si no es cierto.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Qué es lo que hace el siguiente script?

```
__author__ = 'Cátedra de AED'

a = int(input('A: '))
b = int(input('B: '))

may = a
if b > a:
    may = b

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

Seleccione una:

 a.

Lanza un error de intérprete: la instrucción condicional debió tener un `else` para iniciar la rama falsa.

 b.

Lanza un error al ejecutar: las variables `a` y `b` no están definidas cuando se ejecuta la condición.

 c.

Determina el mayor entre las variables `a` y `b`, y deja el valor mayor en la variable `may`. ✓

¡Ok! Efectivamente, comienza suponiendo que el mayor es `a` y asigna ese valor en `may`. Si al preguntar si `b > a` la condición es falsa, el mayor era efectivamente `a` y no hay que cambiar el valor de `may`. Pero si la condición es cierta, el mayor entonces es `b`, y entonces sólo en este caso se cambia el valor de `may`.

 d.

Sólo si el valor de `b` es mayor que el de `a`, deja la variable `may` con el valor de `b`. Si el mayor es `a`, la variable `may` queda con valor indefinido.

¡Correcto!

La respuesta correcta es:

Determina el mayor entre las variables `a` y `b`, y deja el valor mayor en la variable `may`.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes es el nombre del sistema de cifrado de mensajes usado por los alemanes en la Segunda Guerra Mundial, para cuyo descifrado el Ing. Thomas Flowers diseñó la famosa máquina Colossus?

Seleccione una:

- a.  
Cesar
- b.  
Enigma
- c.  
Vigenère
- d.  
Lorenz SZ42 ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

Lorenz SZ42

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Cuál es el nombre del matemático, filósofo e informático teórico en cuyo honor la *Association for Computing Machinery (ACM)* instituyó un premio anual con su nombre, considerado como el equivalente al Nobel en las Ciencias de la Computación?

Seleccione una:

a.



James Gosling

b.



Charles Antony Hoare

c.



Adi Shamir

d.



Alan Turing ✓

¡Ok!

¡Correcto!

La respuesta correcta es:



Alan Turing

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

**¿Qué hace el siguiente segmento de programa?**

```
a = int(input('Primer número: '))
b = int(input('Segundo número: '))

if a < b:
    print('El primer número es el menor')
print('El primer número es el mayor')
```

**Seleccione una:** a.

Si  $a$  es mayor que  $b$ , muestra sólo el mensaje "*El primer número es menor*"

 b.

Si  $a$  es mayor que  $b$ , no muestra un ningún mensaje.

 c.

Si  $a$  es menor que  $b$ , muestra un único mensaje: "*El primer número es menor*".

 d.

Si  $a$  es menor que  $b$ , muestra dos mensajes: "*El primer número es menor*" y luego: "*El primer número es mayor*". ✓

¡Ok! Obviamente, lo que hace este script es incorrecto... pero sólo se le pidió indicar qué hacía, y no si su salida era correcta...

**¡Correcto!****La respuesta correcta es:**

Si  $a$  es menor que  $b$ , muestra dos mensajes: "*El primer número es menor*" y luego: "*El primer número es mayor*".

**Pregunta 8**

Correcta

Puntúa 3 sobre 3

¿Qué hace el siguiente script Python?

```
a = int(input('A: '))
b = int(input('B: '))
c = int(input('C: '))
d = int(input('D: '))

m1 = a
if b > a:
    m1 = b

m2 = c
if d > c:
    m2 = d

if m1 > m2:
    print(m1)
else:
    print(m2)
```

Seleccione una:



a.

Muestra los cuatro valores a, b, c, d en orden de menor a mayor



b.

Muestra el menor de los valores almacenados en las variables a, b, c, d.



c.

Muestra los dos mayores entre los valores a, b, c, d.



d.

Muestra el mayor de los valores almacenados en las variables a, b, c, d.



¡Ok!

¡Correcto!

La respuesta correcta es:

Muestra el mayor de los valores almacenados en las variables a, b, c, d.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

Analice el siguiente script básico en Python:

```
z = 3  
x = 0  
x = x + 1  
x = x + 3  
x = x + z  
print('x:', x)
```

¿Cuál es el valor final almacenado en la variable x que será mostrado al ejecutar el script anterior?

Seleccione una:

- a.  
El valor 7. ✓  
¡Ok!
- b.  
El valor 6.
- c.  
El valor 3.
- d.  
El valor 0.

¡Correcto!

La respuesta correcta es:

El valor 7.

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

Las instrucciones que siguen son expresiones de acumulación o conteo en Python, escritas en la forma general. Seleccione para cada una de ellas, su correspondiente forma resumida:

`x = x - 5*d``x -= 5*d``x = x + p``x += p``x = x + pow(y, 0.5)``x += pow(y, 0.5)``x = x % 2``x %= 2``x = x / 3``x /= 3``x = x * 6``x *= 6`**¡Ok!**

La respuesta correcta es:  $x = x - 5*d \rightarrow x -= 5*d$ ,  $x = x + p \rightarrow x += p$ ,  $x = x + pow(y, 0.5) \rightarrow x += pow(y, 0.5)$ ,  $x = x \% 2 \rightarrow x %= 2$ ,  $x = x / 3 \rightarrow x /= 3$ ,  $x = x * 6 \rightarrow x *= 6$

---

**Comenzado el** martes, 15 de mayo de 2018, 21:54

**Estado** Finalizado

**Finalizado en** martes, 15 de mayo de 2018, 22:20

**Tiempo empleado** 26 minutos

**Puntos** 16/16

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Qué hace el siguiente programa, tal como está escrito, en Python?

```
__author__ = 'Catedra de AED'

c = 0
while c < 3:
    c += 1
    print('Vuelta numero:', c)
```

Seleccione una:

 a.

Muestra un único mensaje: Vuelta número: 3 ✓

¡Correcto! El ciclo en cada vuelta sólo ejecuta la instrucción `c += 1`, ya que su bloque de acciones sólo contiene a esa instrucción. La salida por pantalla está FUERA del bloque, ya que está indentada fuera de este, y se ejecuta sólo al terminar el ciclo, una única vez... ¿Comprende ahora la importancia de respetar la indentación al escribir un programa en Python?

 b.

Provoca un ciclo infinito.

 c.

No llega a mostrar nada: el ciclo corta sin entrar.

 d.

Muestra tres mensajes en líneas separadas, con vueltas numeradas del 1 al 3.

¡Correcto!

La respuesta correcta es:

Muestra un único mensaje: Vuelta número: 3

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Qué efecto provoca el siguiente programa?

```
__author__ = 'Catedra de AED'

c = 0
while c <= 100:
    print('Vuelta numero:', c)
    c -= 1

print('Programa terminado...')
```

Seleccione una:

- a.  
La aparición del mensaje "*Vuelta numero 0*" y luego el mensaje "*Programa Terminado*" (el ciclo hace sólo una repetición).

- b.  
Un ciclo infinito. ✓

¡Correcto! La variable *c* disminuye su valor en cada vuelta, por lo cual se aleja cada vez más del valor 101 que produciría el corte del ciclo.

- c.  
La aparición del mensaje "*Programa Terminado*" (el ciclo no hace ninguna repetición).
- d.  
Un ciclo de 100 repeticiones con 100 mensajes, y al final el mensaje "*Programa Terminado*".

¡Correcto!

La respuesta correcta es:

Un ciclo infinito.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

Supongamos que se desea mostrar en consola estándar los primeros  $n$  números impares. ¿Es correcto el siguiente script Python? (para simplificar, asuma que el usuario cargará un valor no negativo en  $n$ ):

```
c = 1
n = int(input('Cuántos impares quiere?: '))
while c <= 2*n:
    print(c)
    c += 2
```

Seleccione una:

a.

No. El script mostrado produce un ciclo infinito, ya que la instrucción **c +=2** está fuera del bloque del ciclo. ✓

¡Correcto!

b.

No. El script muestra **2\*n** números impares, en lugar de sólo **n**.

c.

Sí. El script es correcto.

d.

No. El script mostrado muestra números pares en lugar de impares (ya que el contador **c** avanza de a 2 en 2)

¡Correcto!

La respuesta correcta es:

No. El script mostrado produce un ciclo infinito, ya que la instrucción **c +=2** está fuera del bloque del ciclo.

**Pregunta 4**

Correcta

Puntúa 2 sobre 2

Suponga que se pide cargar por teclado un conjunto de números, sin saber cuántos son. Se sabe que al cargar el número 0 se termina la carga de datos. Y se pide calcular el promedio entero de los números cargados. ¿Es correcto el planteo del siguiente programa? (Aclaración: el programa hace lo pedido... pero queremos saber si el planteo es óptimo, o está haciendo algo que no debería (aunque llegue de todos modos al resultado correcto) y/o podría mejorarse en alguna forma...)

```
__author__ = 'Catedra de AED'

def test():
    c = a = p = 0

    x = int(input('Ingrese un numero (con cero termina el
proceso): '))
    while x != 0:
        c += 1
        a += x
        p = a // c
        x = int(input('Ingrese otro (con cero termina el p
roceso): '))

    print('El promedio es:', p)

# script principal...
test()
```

Seleccione una:

a.

El programa calcula el valor correcto del promedio, pero la división debería hacerse fuera del ciclo, al terminar el mismo. ✓

¡Correcto! Así como está hecho, sólo la última división realizada en la última vuelta dará el promedio final... por lo que no hay ninguna necesidad de perder tiempo dividiendo en cada vuelta los valores parciales del acumulador y el contador.

b.

El programa no sólo no calcula correctamente el promedio, sino que además se interrumpe en forma inesperada pues en la primera vuelta del ciclo divide por cero.

c.

El programa no calcula correctamente el promedio: la división está al revés.

d.

El programa calcula correctamente el promedio, y no hay necesidad de ninguna mejora.

¡Correcto!

La respuesta correcta es:

El programa calcula el valor correcto del promedio, pero la división debería hacerse fuera del ciclo, al terminar el mismo.

**Pregunta 5**

Correcta

Puntúa 3 sobre 3

Suponga que se le pide plantear un programa para cargar una secuencia de números hasta que aparezca el 0, y determinar si esa secuencia está ordenada de menor a mayor o no. Una forma correcta de hacerlo, se en el siguiente programa:

```
ok = True
num = int(input('Cargue el primer valor (con 0 corta): '))
anterior = num

while num != 0:
    if num < anterior:
        ok = False
    anterior = num
    num = int(input('Cargue el siguiente valor (con 0 corta): '))

if ok == True:
    print('La secuencia está ordenada...')
else:
    print('La secuencia no está ordenada')
```

Suponga que se modifica el modelo anterior y se reemplaza por el que se muestra más abajo. ¿Funciona correctamente el nuevo esquema?

```
ok = True
num = int(input('Cargue el primer valor (con 0 corta): '))
anterior = num

while num != 0:
    if num < anterior:
        ok = False
    else:
        ok = True
    anterior = num
    num = int(input('Cargue el siguiente valor (con 0 corta): '))

if ok == True:
    print('La secuencia está ordenada...')
else:
    print('La secuencia no está ordenada')
```

Seleccione una:

a.

Sí. Funciona correctamente con el cambio propuesto, aunque ya funcionaba también sin ese cambio. Por lo tanto, el cambio es redundante.

b.

No. No funciona correctamente. Con el cambio sugerido aquí, la función determina si la secuencia está ordenada pero de mayor a menor en lugar de hacerlo de menor a mayor.

c.

No. No funciona correctamente. De hecho, el cambio sugerido provoca un error de intérprete y el programa lanza un error y se interrumpe si en la primera vuelta del ciclo entra por el else de esa condición.

d.

No. No funciona correctamente. Si secuencia fuese de la forma {2, 4, 6, 5, 7, 9} entonces al ingresar el 5 el flag *ok* cambiará a *False* (correctamente) pero luego al ingresar el 7 volverá a cambiar a *True* (incorrectamente). Una vez que el *flag* cambia *False*, no debe volver a *True* durante el resto de la carga. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

No. No funciona correctamente. Si secuencia fuese de la forma {2, 4, 6, 5, 7, 9} entonces al ingresar el 5 el flag *ok* cambiará a *False* (correctamente) pero luego al ingresar el 7 volverá a cambiar a *True* (incorrectamente). Una vez que el *flag* cambia *False*, no debe volver a *True* durante el resto de la carga.

**Pregunta 6**

Correcta

Puntúa 2 sobre 2

El siguiente es el programa original que se presentó en la Ficha 6 para resolver el problema número 15 (cargar por doble lectura una secuencia de datos hasta que aparezca el cero, y calcular cuántos de esos datos estaban en ciertos intervalos, calcular además la suma de todos los datos, y finalmente indicar si alguno de los datos era igual a cero):

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

# inicialización de contadores, acumuladores y banderas...
c1, c2, c3, t = 0, 0, 0, 0
ok = False

# proceso de carga por doble lectura...
# ... hacer la primera lectura...
cant = int(input('Ingrese cantidad vendida (con -1 termina el
proceso): '))
while cant != -1:

    # punto a): chequear en qué intervalo está cada cantidad y
    contar...
    if 0 <= cant < 10000:
        c1 += 1
    elif 10000 <= cant < 15000:
        c2 += 1
    else:
        c3 += 1

    # punto b): acumular cada cantidad...
    t += cant

    # punto c): chequear si cant es 0, y marcar con un flag...
    if cant == 0:
        ok = True

    # hacer la segunda lectura...
    cant = int(input('Ingrese otra cantidad (con -1 termina):
')))

# visualización de resultados... punto a)
print()
print('Cantidad de valores >= 0 pero < 10000:', c1)
print('Cantidad de valores >= 10000 pero < 15000:', c2)
print('Cantidad de valores >= 15000:', c3)

# visualización de resultados... punto b)
print('Cantidad total de vehículos vendidos:', t)

# visualización de resultados... punto c)
# recuerde que lo que sigue es equivalente a if ok == True:
if ok:
    print('Se registró al menos una cantidad de ventas igual c
ero')
else:
    print('No se registró ninguna cantidad de ventas igual cer
o')
```

Y el programa que sigue es una modificación del anterior, de forma que se ha eliminado la **segunda lectura** en la carga de datos. ¿Cuál es el efecto que tiene la eliminación de la **segunda lectura** en el programa modificado?

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

# inicialización de contadores, acumuladores y banderas...
c1, c2, c3, t = 0, 0, 0, 0
ok = False

# proceso de carga por doble lectura...
# ... hacer la primera lectura...
cant = int(input('Ingrese cantidad vendida (con -1 termina el
    proceso): '))
while cant != -1:

    # punto a): chequear en qué intervalo está cada cantidad y
    contar...
    if 0 <= cant < 10000:
        c1 += 1
    elif 10000 <= cant < 15000:
        c2 += 1
    else:
        c3 += 1

    # punto b): acumular cada cantidad...
    t += cant

    # punto c): chequear si cant es 0, y marcar con un flag...
    if cant == 0:
        ok = True

# visualización de resultados... punto a)
print()
print('Cantidad de valores >= 0 pero < 10000:', c1)
print('Cantidad de valores >= 10000 pero < 15000:', c2)
print('Cantidad de valores >= 15000:', c3)

# visualización de resultados... punto b)
print('Cantidad total de vehículos vendidos:', t)

# visualización de resultados... punto c)
# recuerde que lo que sigue es equivalente a if ok == True:
if ok:
    print('Se registró al menos una cantidad de ventas igual a
    cero')
else:
    print('No se registró ninguna cantidad de ventas igual a
    cero')
```

Seleccione una:

a.

Si el primer dato que se cargue al ejecutarlo es el valor -1, el programa se detendrá normalmente y mostrará todos los resultados con sus valores iniciales. Pero si primer dato que se cargue no es -1, entonces el programa entrará en ciclo infinito. ✓

¡Correcto! Así como está hecho, al terminar de cargar el primer número (si este no es -1) parecerá que el programa no está haciendo nada (aunque de hecho estará girando una y otra vez con los mismos datos), pero tampoco pedirá ningún otro número.

b.

Si el primer dato que se cargue al ejecutarlo es el valor -1, el programa se detendrá normalmente y mostrará todos los resultados con sus valores iniciales. Pero si primer dato que se cargue no es -1, entonces el programa pedirá solamente la carga de un nuevo valor, y luego se detendrá mostrando los resultados de acuerdo a los dos únicos datos que aceptó.

c.

No produce ningún efecto especial. El nuevo programa funciona tan bien como el original.

d.

Si el primer dato que se cargue al ejecutarlo es el valor -1, el programa entrará en ciclo infinito. Pero si primer dato que se cargue no es -1, entonces el programa funcionará correctamente

¡Correcto!

La respuesta correcta es:

Si el primer dato que se cargue al ejecutarlo es el valor -1, el programa se detendrá normalmente y mostrará todos los resultados con sus valores iniciales. Pero si primer dato que se cargue no es -1, entonces el programa entrará en ciclo infinito.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

El siguiente es **OTRA VEZ** el programa original que se presentó en la Ficha 6 para resolver el problema número 15 (cargar por doble lectura una secuencia de datos hasta que aparezca el cero, y calcular cuántos de esos datos estaban en ciertos intervalos, calcular además la suma de todos los datos, y finalmente indicar si alguno de los datos era igual a cero):

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

# inicialización de contadores, acumuladores y banderas...
c1, c2, c3, t = 0, 0, 0, 0
ok = False

# proceso de carga por doble lectura...
# ... hacer la primera lectura...
cant = int(input('Ingrese cantidad vendida (con -1 termina el
proceso): '))
while cant != -1:

    # punto a): chequear en qué intervalo está cada cantidad y
    contar...
    if 0 <= cant < 10000:
        c1 += 1
    elif 10000 <= cant < 15000:
        c2 += 1
    else:
        c3 += 1

    # punto b): acumular cada cantidad...
    t += cant

    # punto c): chequear si cant es 0, y marcar con un flag...
    if cant == 0:
        ok = True

    # hacer la segunda lectura...
    cant = int(input('Ingrese otra cantidad (con -1 termina):
')))

# visualización de resultados... punto a)
print()
print('Cantidad de valores >= 0 pero < 10000:', c1)
print('Cantidad de valores >= 10000 pero < 15000:', c2)
print('Cantidad de valores >= 15000:', c3)

# visualización de resultados... punto b)
print('Cantidad total de vehículos vendidos:', t)

# visualización de resultados... punto c)
# recuerde que lo que sigue es equivalente a if ok == True:
if ok:
    print('Se registró al menos una cantidad de ventas igual c
ero')
else:
    print('No se registró ninguna cantidad de ventas igual cer
o')
```

Y el programa que sigue es una modificación del anterior, de forma que se ha eliminado la **primera lectura** en la carga de datos. ¿Cuál es el efecto que tiene en Python la eliminación de la **primera lectura** en el programa modificado dejándolo tal cual se ve a continuación?

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

# inicialización de contadores, acumuladores y banderas...
c1, c2, c3, t = 0, 0, 0, 0
ok = False

# proceso de carga por doble lectura...

while cant != -1:

    # punto a): chequear en qué intervalo está cada cantidad y
    contar...
    if 0 <= cant < 10000:
        c1 += 1
    elif 10000 <= cant < 15000:
        c2 += 1
    else:
        c3 += 1

    # punto b): acumular cada cantidad...
    t += cant

    # punto c): chequear si cant es 0, y marcar con un flag...
    if cant == 0:
        ok = True

    # hacer la segunda lectura...
    cant = int(input('Ingrese otra cantidad (con -1 termina):'))


# visualización de resultados... punto a)
print()
print('Cantidad de valores >= 0 pero < 10000:', c1)
print('Cantidad de valores >= 10000 pero < 15000:', c2)
print('Cantidad de valores >= 15000:', c3)

# visualización de resultados... punto b)
print('Cantidad total de vehículos vendidos:', t)

# visualización de resultados... punto c)
# recuerde que lo que sigue es equivalente a if ok == True:
if ok:
    print('Se registró al menos una cantidad de ventas igual cer
o')
else:
    print('No se registró ninguna cantidad de ventas igual cero')
```

Seleccione una:

a.

No produce ningún efecto especial. El nuevo programa funciona tan bien como el original.

b.

Si el primer dato que se cargue al ejecutarlo es el valor -1, el programa se detendrá normalmente y mostrará todos los resultados con sus valores iniciales. Pero si primer dato que se cargue no es -1, entonces el programa pedirá solamente la carga de un nuevo valor, y luego se detendrá mostrando los resultados de acuerdo a los dos únicos datos que aceptó.

c.

Al intentar ejecutar el programa modificado, se producirá un error de intérprete al hacer la comprobación del ciclo while, ya que así como está planteado la variable *cant* quedará indefinida al hacer la primera comprobación. ✓

¡Correcto! Así como está hecho, la variable *cant* nunca llega a ser definida antes de usarse en la primera comprobación del ciclo...

d.

Si el primer dato que se cargue al ejecutarlo es el valor -1, el programa entrará en ciclo infinito. Pero si primer dato que se cargue no es -1, entonces el programa funcionará correctamente

¡Correcto!

La respuesta correcta es:

Al intentar ejecutar el programa modificado, se producirá un error de intérprete al hacer la comprobación del ciclo while, ya que así como está planteado la variable *cant* quedará indefinida al hacer la primera comprobación.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Analice el siguiente script básico en Python:

```
x = -9  
y = 2 + x**0.5  
print(y)
```

¿Cuál es el efecto de ejecutar el script anterior?

Seleccione una:

 a.

Al ejecutarse se interrumpirá en forma abrupta, lanzando un mensaje de operación matemática incorrecta (raíz cuadrada de un número negativo).

 b.

Se ejecutará sin interrumpirse ni lanzar error, y la variable **y** quedará asignada con el valor **5.0** de tipo **float**.

 c.

Se ejecutará sin interrumpirse ni lanzar error, pero la variable **y** quedará asignada con el valor **None** debido a que no puede calcular la raíz cuadrada de un número negativo.

 d.

Se ejecutará sin interrumpirse ni lanzar error, y la variable **y** quedará asignada con el valor **(2+3j)** de tipo **complex**. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

Se ejecutará sin interrumpirse ni lanzar error, y la variable **y** quedará asignada con el valor **(2+3j)** de tipo **complex**.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

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.

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

¡Ok!

c.

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

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).

¡Correcto!

La respuesta correcta es:

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

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

El siguiente es el programa publicado en la Ficha 6 que calcula las raíces de la ecuación de segundo grado en forma directa, dejando que Python se encargue de los detalles de manejo de los números complejos si fuese el caso:

```
__author__ = 'Cátedra de AED'

seguir = 's'
while seguir == 's' or seguir == 'S':

    # título y carga de datos...
    print('Raíces de la ecuación de segundo grado...')
    a = float(input('a: '))
    b = float(input('b: '))
    c = float(input('c: '))

    # procesos: aplicar directamente las fórmulas...
    x1 = (-b + (b**2 - 4*a*c)**0.5) / (2*a)
    x2 = (-b - (b**2 - 4*a*c)**0.5) / (2*a)

    # visualización de resultados...
    print('x1:', x1)
    print('x2:', x2)

    seguir = input('Desea resolver otra ecuación? (s/n): ')

print('Gracias por utilizar nuestro programa...')
```

La consigna en esta pregunta es que se tome un tiempo para ejecutar y probar este programa, entender lo que ocurre, y simplemente registrar los resultados más abajo. En la columna de la izquierda figuran distintas ecuaciones de segundo grado, y en la columna de la derecha figuran las posibles raíces de cada una, en forma desordenada. Seleccione la solución correcta para cada ecuación (Nota: los decimales a la derecha del punto pueden aparecer redondeados o truncados. Seleccione la respuesta más aproximada si ese fuese el caso).

$3x^2 + 2b + 6 = 0$	x1 = (-0.33333+1.37436j) x2 = (-0.33333-1.37436j)
$x^2 + b + 1 = 0$	x1 = (-0.5+0.86602j) x2 = (-0.5-0.86602j)
$2x^2 = 0$	x1 = 0 x2 = 0
$x^2 - 2b - 3 = 0$	x1 = 3.0 x2 = -1.0
$2x^2 + 4b + 2 = 0$	x1 = -1.0 x2 = -1.0

$$\begin{aligned}-4x^2 + 3b + 1 \\ = 0\end{aligned}$$

$$x1 = -0.25 \quad x2 = 1.0$$



▼

¡Ok!

La respuesta correcta es:

$$3x^2 + 2b + 6 = 0 \rightarrow x1 = (-0.33333+1.37436j) \quad x2 = (-0.33333-1.37436j),$$

$$x^2 + b + 1 = 0 \rightarrow x1 = (-0.5+0.86602j) \quad x2 = (-0.5-0.86602j), \quad 2x^2 = 0 \rightarrow x1 = 0 \quad x2 = 0,$$

$$x^2 - 2b - 3 = 0$$

$$\rightarrow x1 = 3.0 \quad x2 = -1.0, \quad 2x^2 + 4b + 2 = 0 \rightarrow x1 = -1.0 \quad x2 = -1.0,$$

$$-4x^2 + 3b + 1 = 0 \rightarrow x1 = -0.25 \quad x2 = 1.0$$

**Comenzado el** martes, 15 de mayo de 2018, 22:20

**Estado** Finalizado

**Finalizado en** viernes, 25 de mayo de 2018, 11:57

**Tiempo empleado** 9 días 13 horas

**Puntos** 17/17

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Cuál de las siguientes cabeceras de ciclo `for` en Python, producirá un esquema de exactamente  $n$  repeticiones, suponiendo que la variable  $n$  tiene un valor entero mayor a cero previamente asignado?

Seleccione una:

a.

`for i in range(0, n//2):`

b.

`for i in range(n+1):`

c.

`for i in range(n, 0, -1):` ✓

¡Correcto! Las  $n$  repeticiones se hacen mediante una cuenta regresiva desde  $n$  hasta 1...

d.

`for i in range(1, n):`

¡Correcto!

La respuesta correcta es:

`for i in range(n, 0, -1):`

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes scripts **NO** creará una tupla conteniendo exclusivamente los caracteres de la palabra 'Python', en ese mismo orden? (Repetimos: la pregunta es cuál de todos **NO** creará la tupla pedida...)

Seleccione una:

a.

t3 = ('P', 'y', 't', 'h', 'o', 'n')

b.

Python = 'Java'  
t5 = tuple(Python) ✓

¡Ok!

c.

t1 = tuple('Python')

d.

t4 = ()  
for c in 'Python':  
 t4 += c,

e.

t2 = 'P', 'y', 't', 'h', 'o', 'n'

¡Correcto!

La respuesta correcta es:

Python = 'Java'  
t5 = tuple(Python)

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuáles de los siguientes ciclos for mostrarán en pantalla **sólo** números negativos? (Aclaración: más de una respuesta puede ser correcta... marque TODAS las que considere válidas...)

Seleccione una o más de una:

- a. **for** i **in** range(10, 1, -2):  
print('i:', i)
- b. **for** i **in** range(5, -15, -5):  
print('i:', i)
- c. **for** i **in** range(-10, -1):  
print('i:', i) ✓ ¡Ok!
- d. **for** i **in** range(-1, -10, -1):  
print('i:', i) ✓ ¡Ok!

¡Correcto!

Las respuestas correctas son: **for** i **in** range(-1, -10, -1):

```
print('i:', i), for i in range(-10, -1):  
print('i:', i)
```

**Pregunta 4**

Correcta

Puntúa 2 sobre 2

El proceso de búsqueda del mayor en el siguiente programa, contiene una ligera modificación a las variantes analizadas en la Ficha 7: la variable `may` se inicializa en 0 antes del ciclo, y luego simplemente se aplica la idea de comparar el número cargado `num` contra `may` en cada vuelta, sin preocuparse por el valor de `may` frente al primer dato. ¿Hay algún problema en el planteo de esta estrategia?

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante:
      may inicializada en 0)...)
n = int(input('Ingrese n: '))

may = 0
for i in range(1, n+1):
    num = int(input('Numero: '))
    if num > may:
        may = num

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

Seleccione una:

a.

No. No hay ningún problema. Funcionará correctamente en todos los casos.

b.

Sí, hay un problema: si todos los números del conjunto de entrada fuesen cero, se mostraría un `None` como resultado, en lugar de lo que correspondería retornar que es un 0.

c.

Sí, hay un problema: si todos los números del conjunto de entrada fuesen positivos o cero, se mostraría un 0 como resultado, en lugar del mayor del conjunto cargado.

d.

Sí, hay un problema: si todos los números del conjunto de entrada fuesen negativos, se mostraría un 0 como resultado, en lugar del mayor de los negativos cargados. ✓

¡Correcto!

**¡Correcto!**

La respuesta correcta es:

Sí, hay un problema: si todos los números del conjunto de entrada fuesen negativos, se mostraría un 0 como resultado, en lugar del mayor de los negativos cargados.

**Pregunta 5**

Correcta

Puntúa 3 sobre 3

El proceso de búsqueda del mayor en el siguiente programa está ligeramente cambiado con relación a su versión original (la tercera variante del problema de la búsqueda del mayor (Ficha 7): En la versión original, se comenzaba definiendo la variable *may* con el valor *None*, y ahora en esta versión que proponemos, hemos eliminado esa inicialización marcándola como un comentario (y sin borrarla, pero sólo por razones de claridad). ¿Hay algún problema en el planteo de esta pequeña variante? (Recuerde: si la instrucción está marcada como comentario, es lo mismo que si no estuviese).

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante 3)...')

# may = None
b = False
num = int(input('Ingrese un numero (con 0 finaliza): '))
while num != 0:
    if b == False:
        may = num
        b = True
    elif num > may:
        may = num
    num = int(input('Ingrese otro (con 0 finaliza): '))

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

Seleccione una:

a.

No. No hay ningún problema. Funcionará correctamente en todos los casos, y la instrucción *may = None* era completamente innecesaria en la versión original.

b.

Sí, hay un problema: si en la primera carga antes del ciclo se ingresa un 0, el ciclo no ejecutará su bloque de acciones y la variable *may* quedará sin definir, provocando en ese caso que el programa se interrumpa y lance un error al intentar mostrar el valor de *may* en el programa. ✓

¡Correcto!

c.

Sí, hay un problema: si todos los números que cargan fuesen negativos, en ese caso la variable *may* por defecto quedaría valiendo 0, y el programa mostraría un cero al final.

d.

Sí, hay un problema: la variable *may* estará sin definir incluso dentro del ciclo. Cuando se intente asignar en ella el valor de *num* que corresponda a cada vuelta, el programa se interrumpirá y lanzará un error de variable no definida.

### ¡Correcto!

La respuesta correcta es:

Sí, hay un problema: si en la primera carga antes del ciclo se ingresa un 0, el ciclo no ejecutará su bloque de acciones y la variable *may* quedará sin definir, provocando en ese caso que el programa se interrumpa y lance un error al intentar mostrar el valor de *may* en el programa.

**Pregunta 6**

Correcta

Puntúa 3 sobre 3

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.

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.

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.

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

d.

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.

### ¡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 7**

Correcta

Puntúa 3 sobre 3

Supongamos que se nos pide buscar el mayor de una secuencia de  $n$  números, tal como se analizó en la Ficha 7, pero de modo que ahora además de mostrar el mayor, se muestre también en qué orden apareció en la carga (o sea, en qué vuelta del ciclo apareció ese mayor) Por ejemplo, si la secuencia a cargar fuese  $\{2, 5, 1, 3\}$  entonces el mayor sería el 5 y apareció en el lugar 2 (o en el orden 2, o en la vuelta 2 del ciclo) ¿Está bien planteado el siguiente programa para cumplir con este nuevo requerimiento?

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante 1)...')

n = int(input('n: '))
for v in range(1, n+1):
    num = int(input('Numero: '))
    if v == 1:
        may, pos = num, 1
    elif num > may:
        may, pos = num, v

print('El mayor es:', may)
print('Se cargó en la vuelta:', pos)
```

Seleccione una:

a.

Sí, está correctamente planteado. ✓

¡Correcto!

b.

No. No está bien planteado: si el valor mayor apareciese repetido más de una vez en la carga, la variable *pos* quedaría valiendo *None*.

c.

No. No está bien planteado: siempre muestra que la vuelta en que se cargó el mayor fue la 1 del ciclo.

d.

No. No está bien planteado: los resultados se están mostrando al revés, ya que el mayor estaría en *pos* y su posición de carga estaría en *may*. Hay que invertir las asignaciones de las tuplas dentro del ciclo.

¡Correcto!

La respuesta correcta es:

Sí, está correctamente planteado.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Analice el siguiente script en el cual se recorre una secuencia llamada sec con un ciclo *for*:

```
# suponga que sec es una secuencia (cadena,  
tupla, rango, etc.)  
# ya inicializada  
for x in sec:  
    print('x:', x)
```

¿Qué hace este script si la secuencia sec estuviese inicialmente vacía?

Seleccione una:

a.

El ciclo *for* no hace ninguna repetición y el script termina normalmente sin mostrar nada en pantalla. ✓

¡Ok! Efectivamente... el ciclo for es de tipo [0, N] por lo que si la secuencia a recorrer está vacía, el for no llega a hacer ninguna repetición y sigue de largo a buscar la siguiente instrucción.

b.

El ciclo *for* hace una y sólo una repetición y muestra el mensaje x: None.

c.

El ciclo *for* entra en repetición infinita.

d.

El ciclo *for* no hace ninguna repetición pero el script se interrumpe lanzando un mensaje de error.

¡Correcto!

La respuesta correcta es:

El ciclo *for* no hace ninguna repetición y el script termina normalmente sin mostrar nada en pantalla.

**Pregunta 9**

Correcta

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.

Python provee dos tipos de ciclos básicos: el *while* y el *for*. ✓

¡Ok!

c.

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*.

d.

Todos los ciclos básicos de Python, son de la forma [0, N]. ✓

¡Ok!

¡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

Puntúa 1 sobre 1

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.

$t \geq -70 \text{ or } t \leq 50$

b.

$t < -70 \text{ and } t > 50$

c.

$t \geq -70 \text{ and } t \leq 50$

d.

$t < -70 \text{ or } t > 50$  ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

$t < -70 \text{ or } t > 50$

**Comenzado el** martes, 15 de mayo de 2018, 22:32

**Estado** Finalizado

**Finalizado en** viernes, 18 de mayo de 2018, 08:06

**Tiempo empleado** 2 días 9 horas

**Puntos** 17/17

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 2 sobre 2

El siguiente programa es una variante del que mostramos en la [Ficha 7](#) para resolver el problema 20. En esta variante, hemos modificado la lógica para que al buscar el menor importe del vendedor 2 no se tenga que preguntar por el primer dato en cada vuelta, y eliminar la bandera que se aplicaba en la versión original. ¿Hay algún problema en el planteo de esta variante? Seleccione la respuesta que mejor describa lo que está pasando con el programa propuesto.

```
_author_ = 'Catedra de AED'

# pasó la primera venta del vendedor 2?
aviso = False

# si no se cargan ventas del vendedor 2, menor_importe queda en None...
menor_importe = None

# acumuladores de cantidades...
c1 = c2 = 0

# acumuladores de importes...
i1 = i2 = 0

print('Ventas de un Comercio... ingrese los datos de cada venta...')

# ingresar datos de la primera venta...
codigo = -1
while codigo < 0 or codigo > 2:
    codigo = int(input('Codigo de vendedor (1 o 2) (0 para cortar): '))
    if codigo > 2 or codigo < 0:
        print('Error... se pidió 1 o 2 o 0 para cortar...')

cantidad = int(input('Cantidad vendida: '))
importe = float(input('Importe: '))

menor_importe = importe

while codigo != 0:
    if codigo == 1:
        c1 += cantidad
        i1 += importe

    elif codigo == 2:
        c2 += cantidad
        i2 += importe

    # Aplicar mecanismo de cálculo del menor...
    if importe < menor_importe:
        menor_importe = importe

# ingresar el siguiente código y volver al ciclo...
codigo = -1
while codigo < 0 or codigo > 2:
    codigo = int(input('Codigo de vendedor (1 o 2) (0 para cortar): '))
    if codigo > 2 or codigo < 0:
        print('Error... se pidió 1 o 2 o 0 para cortar...')

cantidad = int(input('Cantidad vendida: '))
importe = float(input('Importe: '))

# Calcular el importe promedio...
promedio = (i1 + i2) / 2

print('Cantidad de productos vendida por el vendedor 1:', c1)
print('Cantidad de productos vendida por el vendedor 2:', c2)
print('Importe total facturado por el vendedor 1:', i1)
print('Importe total facturado por el vendedor 2:', i2)
print('Importe de la menor venta del vendedor 2:', menor_importe)
print('Importe promedio entre los dos vendedores:', promedio)
```

Seleccione una:

- a.  
Sí. Hay un único problema: los tres datos de la venta se cargan juntos. Si código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido.
- b.  
No. No hay ningún problema. Funcionará correctamente en todos los casos.
- c.  
Sí. Hay un único problema: se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor\_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.
- d.  
Sí. De hecho, hay dos problemas: el primero es que los tres datos de la venta se cargan juntos. Si el código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido. Y el segundo problema (y más grave) es que se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor\_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.

✓  
¡Correcto!

#### ¡Correcto!

La respuesta correcta es:

Sí. De hecho, hay dos problemas: el primero es que los tres datos de la venta se cargan juntos. Si el código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido. Y el segundo problema (y más grave) es que se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor\_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.

**Pregunta 2**

Correcta

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) ▾

Esquema  
en el cual  
se  
desconoce  
de  
antemano  
el número  
de  
repeticiones

Ciclo while (operando en forma natural como [0, n]) ▾

Esquema  
en el cual  
se sabe de  
antemano  
el número  
de  
repeticiones

Ciclo for (iterando sobre un range cuyo tamaño coincide 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) ▾

Recorrido  
de una  
secuencia  
(tupla,  
range,  
cadena,  
lista, etc.)

Ciclo for (iterando sobre la estructura de datos a procesar) ▾

¡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), Esquema en el cual se desconoce de antemano el número de repeticiones → Ciclo while (operando en forma natural como [0, n]), Esquema en el cual se sabe de antemano el número de repeticiones → Ciclo for (iterando sobre un range cuyo tamaño coincide 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), Recorrido de una secuencia (tupla, range, cadena, lista, etc.) → Ciclo for (iterando sobre la estructura de datos a procesar)

**Pregunta 3**

Correcta

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.

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!



c.

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!



d.

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.

¡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 4**

Correcta

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 un número 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 un número 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.

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

¡Correcto!

b.

El script 1 es incorrecto (mostrará el mensaje "El dato cargado fue validado en forma correcta..." muchas veces) y el segundo es correcto.

c.

El script 1 es correcto, pero el segundo no lo es (nunca mostrará el mensaje "El dato cargado fue validado en forma correcta...")

d.

Ambos son incorrectos.

¡Correcto!

La respuesta correcta es:

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

**Pregunta 5**

Correcta

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 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.

b.

No. No hay ningún error en el programa.

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.

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.

e.

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. ✓

¡Correcto!

**¡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 6**

Correcta

Puntúa 1 sobre 1

¿Cuál es el significado de la sigla *IGU* (en español) en el ámbito del diseño de pantallas y sistema de carga de datos de un programa?

Seleccione una:

- a.  
Implementación Guiada por el Usuario
- b.  
Interfaz Gráfica de Usuario ✓  
¡Ok!
- c.  
Interfaz Genuina de Usuario
- d.  
Independencia Gráfica Universal

¡Correcto!

La respuesta correcta es:

Interfaz Gráfica de Usuario

**Pregunta 7**

Correcta

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 empleando 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: '))

# límites 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.

Sí. Funciona correctamente para todos los casos.

b.

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. ✓

¡Correcto!

c.

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.

d.

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.

### ¡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 8

Correcta

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('Ingrese 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 '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.

b.

No hay ningún problema.

c.

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'.

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 9**

Correcta

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 se interrumpirá al ejecutarla, lanzando un mensaje de error, si el número seleccionado por *randint()* es 0.

b.

No causará ningún problema. El programa seguirá funcionando y mostrando sus resultados en forma correcta.

c.

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.

d.

En sí 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. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

En sí 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 10

Correcta

Puntúa 1 sobre 1

¿Qué se entiende por **momento epoch** cuando se trabaja con la función `time.clock()` 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 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.

c.

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.

d.

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!

¡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.

**Comenzado el** domingo, 27 de mayo de 2018, 16:34

**Estado** Finalizado

**Finalizado en** domingo, 27 de mayo de 2018, 17:07

**Tiempo empleado** 32 minutos 51 segundos

**Puntos** 22/22

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes es claramente **FALSA** respecto de la estrategia de dividir un problema en subproblemas?

Seleccione una:

a.

La división en subproblemas garantiza que se encontrará una solución para el problema en estudio. ✓

¡Ok! Lamentablemente, nada garantiza que encontremos una solución para cualquier problema que se nos plante...

b.

La división en subproblemas permite generar subrutinas que luego podrán usarse nuevamente en otros planteos.

c.

La división en subproblemas facilita la comprensión del problema por parte del programador.

d.

La división en subproblemas permite dividir el trabajo entre varios programadores.

¡Correcto!

La respuesta correcta es:

La división en subproblemas garantiza que se encontrará una solución para el problema en estudio.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Qué hace el siguiente programa en Python?

```
__author__ = 'Cátedra de AED'

def comparar():
    if a > p:
        print('El valor', a, 'es mayor...')

    if b > p:
        print('El valor', b, 'es mayor...')

    if c > p:
        print('El valor', c, 'es mayor...')

def promedio():
    global p
    p = (a + b + c) / 3

a = int(input('A: '))
b = int(input('B: '))
c = int(input('C: '))

promedio()
comparar()

print('Programa terminado...')
```

Seleccione una:

 a.

Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra sólo los valores de las variables que sean mayores al promedio. ✓

¡Ok!

 b.

Lanza error de intérprete: las variables *a*, *b*, *c* y *p* están definidas incorrectamente (debieron definirse todas en el script principal, debajo de las funciones).

 c.

Muestra los valores de *a*, *b*, *c*, y también el *promedio* de los valores de esas tres variables.

 d.

Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra los valores de todas las variables.

¡Correcto!

La respuesta correcta es:

Calcula el promedio de los valores de  $a$ ,  $b$  y  $c$ , y luego muestra sólo los valores de las variables que sean mayores al promedio.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

En la columna de la izquierda se describen algunas situaciones de programación que necesitan ser resueltas mediante el desarrollo de alguna función, y se muestra el bloque de acciones de las posibles funciones, sin sus cabeceras. Y en la columna de la derecha, figuran las posibles cabeceras de ese funciones. Para cada situación, seleccione la cabecera que sería adecuada.

Tomar como parámetro una cadena de caracteres y dos números enteros. Retornar los dos caracteres de la cadena que estén ubicados en las posiciones indicadas por los dos números:

# Cabecera??

```
return s[i1], s[i2]
```

def valores(s, i1, i2) ▾



Tomar como parámetro los valores de dos temperaturas, y retornar un valor lógico que indique si ambas son mayores a cero:

# Cabecera??

def chequear(t1, t2): ▾

```
if t1 > 0 and t2 > 0:  
    r = True  
else:  
    r = False  
return r
```



Tomar como parámetro dos números y un valor boolean. Si el boolean es true, calcular y retornar el cociente de los dos números. Si el boolean es false, retornar el producto de ambos números:

# Cabecera??

def proceso(x, y, flag): ▾



```
if flag:  
    r = x / y  
else:  
    r = x * y  
return r
```

Tomar como parámetro las edades de tres personas, y retornar la menor edad:

# Cabecera??

```
if e1 < e2 and e1 < e3:  
    m = e1  
else:  
    if e2 < e3:  
        m = e2  
    else:  
        m = e3  
return m
```

def menor(e1, e2, e3): ▾



¡Ok!

La respuesta correcta es:

Tomar como parámetro una cadena de caracteres y dos números enteros. Retornar los dos caracteres de la cadena que estén ubicados en las posiciones indicadas por los dos números:

```
# Cabecera??  
  
return s[i1], s[i2]
```

→ def valores(s, i1, i2 ),

Tomar como parámetro los valores de dos temperaturas, y retornar un valor lógico que indique si ambas son mayores a cero:

```
# Cabecera??  
  
if t1 > 0 and t2 > 0:  
    r = True  
else:  
    r = False  
return r
```

→ def chequear(t1, t2):,

Tomar como parámetro dos números y un valor boolean. Si el boolean es true, calcular y retornar el cociente de los dos números. Si el boolean es false, retornar el producto de ambos números:

```
# Cabecera??  
  
if flag:  
    r = x / y  
else:           → def proceso(x, y, flag):,  
    r = x * y  
return r
```

Tomar como parámetro las edades de tres personas, y retornar la menor edad:

```
# Cabecera??  
  
if e1 < e2 and e1 < e3:  
    m = e1  
else:  
    if e2 < e3:           → def menor(e1, e2, e3):  
        m = e2  
    else:  
        m = e3  
return m
```

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Suponga que se desea desarrollar un programa que cargue dos números, y muestre el mayor de los números pero también el valor de multiplicar por dos y por tres a ese mayor. ¿Está bien planteado el programa que sigue?

```
__author__ = 'Cátedra de AED'

def cargar():
    a = int(input('A: '))
    b = int(input('B: '))
    return a, b

def comparar(a, b):
    if a > b:
        may = a
    else:
        may = b
    return may

def procesar(may):
    doble = 2 * may
    triple = 3 * may
    return doble, triple

def mostrar(may, doble, triple):
    print('El mayor es:', may)
    print('El doble del mayor es:', doble)
    print('El triple del mayor es:', triple)

# script principal...
a, b = cargar()
doble, triple = procesar(may)
may = comparar(a, b)
mostrar(may, doble, triple)
```

Seleccione una:

a.

Sí. El programa está correctamente planteado.

b.

El programa está mal planteado: la función *comparar()* debería ser invocada antes de invocar a la función *procesar()*. Así como está, lanza un error al intentar ejecutar la función *procesar()* pues no reconoce a la variable *may*.

¡Ok!

c.

El programa está mal planteado: la visualización de los resultados finales DEBE hacerse en el script principal y NO en una función separada.

d.

El programa lanza un error de intérprete: el script principal no puede consistir sólo en llamadas a funciones.

¡Correcto!

La respuesta correcta es:

El programa está mal planteado: la función *comparar()* debería ser invocada antes de invocar a la función *procesar()*. Así como está, lanza un error al intentar ejecutar la función *procesar()* pues no reconoce a la variable *may*.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere desarrollar un programa que cargue por teclado el nombre de una persona, su edad, y su sueldo anual promedio, y que luego se desea invocar a una función llamada **convertir()** que tome como parámetros a esos valores, los procese, y retorne una cadena con todos esos datos convertidos a una cadena de caracteres y concatenados. Tomando como modelo de uso a la función **test()** que se muestra en el programa de mas abajo... ¿cuál de las que se proponen como opciones podría ser la cabecera de la función **convertir()**?

```
__author__ = 'Cátedra de AED'

# Suponga que aquí está definida la función convertir()
# def .....  
# .....  
# .....
```

```
def test():
    nombre = input('Ingrese el nombre: ')
    edad = int(input('Ingrese edad: '))
    sueldo = float(input('Ingrese sueldo: '))

    resultado = convertir(nombre, edad, sueldo)

    print('Datos registrados:', resultado)

# script principal
test()
```

Seleccione una:

a.

def convierta(nombre, edad, sueldo):

b.

def convertir():

c.

def convertir(nombre, edad, sueldo): ✓

¡Ok!

d.

def convertir(edad, sueldo):

¡Correcto!

La respuesta correcta es:

```
def convertir(nombre, edad, sueldo):
```

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere desarrollar un programa para cargar una cadena de caracteres por teclado y mostrar esa cadena en la consola de salida mediante una función. ¿Cuál de las posibles respuestas describe mejor lo que ocurre con el programa que mostramos aquí?

```
__author__ = 'Cátedra de AED'

def mensaje(m):
    print('Mensaje:', m)

def test():
    mens = input('Ingrese un mensaje a mostrar: ')
    r = mensaje(mens)
    print('Programa terminado...')

# script principal...
test()
```

Seleccione una:

a.

El programa compila (el intérprete no lanza error alguno al comenzar a ejecutarlo) y ejecuta, pero no muestra correctamente el mensaje esperado: en su lugar, muestra el valor *None*.

b.

El programa no compila (el intérprete lanza un error de sintaxis antes comenzar a ejecutarlo): No se puede invocar a la función *mensaje()* y asignarla en una variable, ya que esa función es de la forma sin retorno de valor.

c.

El programa funciona y hace exactamente lo esperado, pero la función *mensaje()* es una función sin retorno de valor, por lo que no debería ser asignada en una variable cuando se la invoca. ✓

¡OK!

d.

El programa funciona, hace exactamente lo esperado, y no presenta ningún tipo de inconveniente ni elementos extraños en su código fuente.

¡Correcto!

La respuesta correcta es:

El programa funciona y hace exactamente lo esperado, pero la función *mensaje()* es una función sin retorno de valor, por lo que no debería ser asignada en una variable cuando se la invoca.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere desarrollar un programa para cargar un número por teclado y mostrar el triple de ese número en consola de salida ¿Está bien planteado el siguiente programa?

```
__author__ = 'Cátedra de AED'

def triple(n):
    t = 3 * n

def test():
    a = int(input('Ingrese un numero: '))
    r = triple(a)
    print('El triple del numero es:', r)

# script principal
test()
```

Seleccione una:

a.

El programa lanza un error de intérprete: el parámetro formal de la función *triple()* no puede llamarse *n* (debería llamarse *a*).

b.

El programa ejecuta sin lanzar error, pero muestra un resultado *None* en lugar del triple del número cargado: falta la instrucción *return t* al final de la función *triple()*. ✓

¡Ok!

c.

El programa lanza un error de intérprete: la función *triple()* está definida como una función sin retorno de valor, y se producirá un error al invocarla y asignar su retorno en *t*.

d.

Sí, el programa hace exactamente lo esperado.

¡Correcto!

La respuesta correcta es:

El programa ejecuta sin lanzar error, pero muestra un resultado *None* en lugar del triple del número cargado: falta la instrucción *return t* al final de la función *triple()*.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Qué hace la siguiente función en Python?

```
def check(n):
    if n % 2 == 0:
        return True
    else:
        return False
```

Seleccione una:

 a.

Retorna True si el número n tomado como parámetro es primo.

 b.

Retorna True si el número n tomado como parámetro es par. ✓

¡Ok!

 c.

Retorna True si el número n tomado como parámetro es mayor a 2.

 d.

Retorna True si el número n tomado como parámetro es impar.

¡Correcto!

La respuesta correcta es:

Retorna True si el número n tomado como parámetro es par.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Qué efecto produce la ejecución del siguiente script, tal y como se muestra (exactamente con los valores 3 y 0 indicados como parámetro al invocar a la función `calcular()`)?

```
__author__ = 'Catedra de AED'

def calcular(a, b):
    if b != 0:
        c = a // b
        r = a % b
    return c, r

# script principal
a, b = calcular(3, 0)
print('Cociente:', a, 'Resto:', b)
```

Seleccione una:

- a.  
Ejecuta sin problemas. Muestra el mensaje "Cociente: None Resto: None"
- b.  
Ejecuta sin problemas. Muestra el mensaje "Cociente: 0 Resto: 0"
- c.  
Se produce un error al intentar ejecutar la instrucción `a, b = calcular(3, 0).` ✓  
¡Correcto! La función estaría retornando UN None (y no una tupla), con lo cual la asignación en dos variables para desempaquetar la tupla no es válida. El intérprete lanza un error.
- d.  
Ejecuta sin problemas. Muestra el mensaje "None"

¡Correcto!

La respuesta correcta es:

Se produce un error al intentar ejecutar la instrucción `a, b = calcular(3, 0).`

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

El siguiente programa carga por teclado dos números, usa una función para obtener el mayor entre ambos, y finalmente muestra el mayor. ¿Está bien planteado el programa?

```
__author__ = 'Cátedra de AED'

def comprobar(n1, n2):
    if n1 > n2:
        return n1

def test():
    n1 = int(input('Ingrese un número: '))
    n2 = int(input('Ingrese otro: '))
    r = comprobar(n1, n2)
    print('El mayor es:', r)

# script principal
test()
```

Seleccione una:

a.

No. El programa ejecuta sin problemas pero siempre muestra como mayor al primer número, ya que no queda claro lo que pasa si el mayor fuese el segundo.

b.

No. El programa lanza un error de intérprete en la función comprobar(), ya que no tiene return en la rama falsa.

c.

Sí. Está correctamente planteado

d.

No. El programa muestra el valor None si el mayor es el segundo numero. La función comprobar() Debería tener un return en cada rama lógica que se plantee dentro de ella (falta un return en la rama falsa...)



¡Ok!

¡Correcto!

La respuesta correcta es:

No. El programa muestra el valor None si el mayor es el segundo numero.  
La función comprobar() Debería tener un return en cada rama lógica que se plantee dentro de ella (falta un return en la rama falsa...)

**Pregunta 11**

Correcta

Puntúa 1 sobre 1

El siguiente es un programa simple, que carga por teclado un número y usa una función para chequear si el mismo es cero o positivo (en cuyo caso, avisa con un mensaje) ¿Hay algún problema en el programa mostrado?

```
__author__ = 'Cátedra de AED'

def mostrar(n):
    if n < 0:
        return
    print('El número', n, 'es válido')

def test():
    n = int(input('Ingrese un número: '))
    mostrar(n)
    print('Programa terminado...')

# script principal
test()
```

Seleccione una:

a.

El programa ejecuta sin problemas, pero SIEMPRE muestra el mensaje que indica que el número es válido (debería mostrarlo sólo si el número es mayor o igual a cero).

b.

El programa lanza un error de intérprete: no se puede usar *return* en una función sin indicarle el valor a retornar.

c.

El programa lanza un error de intérprete: una función que usa un *return* sin valor indicado, no puede tomar parámetros.

d.

No hay ningún problema: el programa hace exactamente lo esperado y no tiene elementos extraños en su planteo. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

No hay ningún problema: el programa hace exactamente lo esperado y no tiene elementos extraños en su planteo.

**Pregunta 12**

Correcta

Puntúa 2 sobre 2

Considere el programa para el *Juego del Número Secreto* que se presentó en la Ficha 7. Ese programa se planteó originalmente sin funciones, y aquí lo mostramos redefinido para emplear funciones. Pero además, en la versión original del programa se usaba una *bandera* para marcar en el algoritmo si el número secreto fue encontrado o no; y nos proponemos ahora tratar de eliminar el uso de esa bandera y simplificar la estructura del programa. Sugerimos la modificación que se muestra más abajo empleando una *instrucción return* para cortar el ciclo y la función 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


def play_secret_number_game(limite_derecho, cantidad_intentos):

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

    # contador de intentos...
    intentos = 0

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

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

        # ... 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)
```

```

def test():
    print('Juego del Número Secreto... Configuración Inicial...')
    ld = int(input('El número secreto estará entre 1 y: '))
    ci = int(input('Cantidad máxima de intentos que tendrá disponible: '))
    play_secret_number_game(ld, ci)

# script principal...
test()

```

Seleccione una:

a.

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.

b.

Sí. Funciona correctamente para todos los casos. ✓

¡Correcto!

c.

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.

d.

No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto la función *play\_secret\_number\_game()* mostrará correctamente el mensaje avisando que ganó y cortará el ciclo con la instrucción *return*. Pero luego la función continuará e inmediatamente mostrará también el mensaje avisando que el jugador perdió, provocando ambigüedad.

¡Correcto!

La respuesta correcta es:

Sí. Funciona correctamente para todos los casos.

**Pregunta 13**

Correcta

Puntúa 2 sobre 2

El siguiente programa usa una función para verificar si un número que se carga por teclado es mayor que el triple de otro número que también se carga. ¿Cuál es el efecto de ejecutar esta función?

```
__author__ = 'Catedra de AED'

def ejemplo():
    i = int(input('Ingrese un valor: '))
    t = int(input('Ingrese otro: '))
    if i > 3*t:
        ok = True

    if ok:
        print('El primer valor es mayor al triple del segundo...')

# script principal
ejemplo()
```

Seleccione una:

a.

Mostrará el mensaje *El primer valor es mayor al triple del segundo* sólo si el valor cargado en *i* es menor al valor de *t* multiplicado por 3. No hará nada en caso contrario.

b.

Mostrará el mensaje *El primer valor es mayor al triple del segundo* sólo si el valor cargado en *i* es menor o igual al valor cargado en *t* multiplicado por 3. No hará nada en caso contrario.

c.

Mostrará el mensaje *El primer valor es mayor al triple del segundo* sean cuales sean los valores que se carguen en *i* y en *t*.

d.

Mostrará el mensaje *El primer valor es mayor al triple del segundo* si *i* > *3\*t*, pero lanzará un error y se interrumpirá si *i* <= *3\*t*, ya que en este caso la variable *ok* quedaría sin definir. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

Mostrará el mensaje *El primer valor es mayor al triple del segundo* si *i* > *3\*t*, pero lanzará un error y se interrumpirá si *i* <= *3\*t*, ya que en este caso la variable *ok* quedaría sin definir.

**Pregunta 14**

Correcta

Puntúa 2 sobre 2

¿En cuáles de los siguientes casos el *algoritmo de divisiones sucesivas es efectivamente muy lento e impráctico* para determinar la primalidad de un número  $n$ ? (En otras palabras: ¿qué características debe tener el número  $n$  para que el algoritmo caiga en un *peor caso* o cerca de un *peor caso*?) (Observación: más de una respuesta puede ser válida. Marque todas las que considere correctas)

Seleccione una o más de una:



a.

Que  $n$  sea un número par.



b.

Que  $n$  sea un número impar



c.

Que  $n$  sea un número compuesto (no primo) pero tal que el primero de sus divisores sea a su vez un número muy grande. ✓

**¡Correcto!** En este caso, el algoritmo deberá hacer **muchas** de todas las divisiones posibles... esto no cae necesariamente en el peor caso, pero se aproxima bastante...



d.

Que  $n$  sea ya un número primo muy grande (por ejemplo:  $n > 10^{20}$ ) ✓

**¡Correcto!** En este caso, el algoritmo deberá hacer **todas** las divisiones posibles, que como vimos serán muchas... y caerá en el peor caso posible.

**¡Correcto!**

Las respuestas correctas son:

Que  $n$  sea ya un número primo muy grande (por ejemplo:  $n > 10^{20}$ ),

Que  $n$  sea un número compuesto (no primo) pero tal que el primero de sus divisores sea a su vez un número muy grande.

**Pregunta 15**

Correcta

Puntúa 3 sobre 3

Se tiene un número entero positivo  $n$  de 40 dígitos ( $n \geq 10^{40}$ ) y se quiere determinar si  $n$  es primo aplicando el *algoritmo de divisiones sucesivas* visto en la sección de *Temas Avanzados* de la *Ficha 9* (básicamente, controlando todo divisor posible en el intervalo  $[2.. \sqrt{n}]$ ). Si nuestra computadora ejecuta unas mil millones de divisiones por segundo (1000000000 de divisiones por segundo =  $10^9$  divisiones por segundo), ¿cuánto tiempo le llevaría en el peor caso a esa computadora determinar si  $n$  es primo? (Sugerencia: analice con cuidado la Ficha 9, página 219...)

Seleccione una:

- a.  
Alrededor de una hora.
- b.  
Alrededor de un día.
- c.  
Alrededor de una semana.
- d.  
Alrededor de un año.
- e.  
Alrededor de cien años.
- f.  
Alrededor de mil años.
- g.  
Alrededor de tres mil años. ✓  
¡Correcto! En las condiciones dadas, el cálculo da aproximadamente 3170 años.
- h.  
Alrededor de treinta mil años.
- i.  
Alrededor de trescientos mil años.
- j.  
Alrededor de trescientos millones de años.

**¡Correcto!**

La respuesta correcta es:

Pregunta 16

Correcta

Puntúa 2 sobre 2

¿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:

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. ✓

**¡Correcto!** Basta con poner un ejemplo... si  $n = 24$  la factorización es  $2 * 2 * 2 * 3$  con lo que el factor 2 aparece más de una vez...

c.

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

**¡Correcto!** Si  $n > 2$  contiene al 2 como factor primo, entonces  $n$  es divisible por 2 y por lo tanto no sería impar...

d.

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

**¡Correcto!**

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.

**Comenzado el** sábado, 16 de junio de 2018, 18:40

**Estado** Finalizado

**Finalizado en** domingo, 17 de junio de 2018, 16:43

**Tiempo empleado** 22 horas 3 minutos

**Puntos** 20/21

**Calificación** 10 de 10 (96%)

**Pregunta 1**

Correcta

Puntúa 2 sobre 2

Suponga que se desea un programa que cargue dos números, y muestre el producto y el cociente entre ellos ¿Cuál es el problema con el siguiente programa desarrollado en base a funciones que manejan variables globales? (ver Ficha 09),

```
__author__ = 'Cátedra de AED'

def cargar():
    global a, b
    a = int(input('A: '))
    b = int(input('B: '))

def calcular():
    c = a * b
    d = a / b

# script principal...
cargar()
calcular()

print('Producto:', c)
print('Cociente:', d)
```

Seleccione una:

a.

Lanza un error de intérprete en el mismo inicio: todas las variables debieron ser definidas (asignadas) en el script principal.

b.

El programa lanza un error al ser ejecutado, en el *script principal*: no reconoce las variables *c* y *d* ya que fueron definidas en la función *calcular()*, pero como variables locales a ella. ✓

¡Ok!

c.

No lanza errores de intérprete, pero funciona mal: en el *script principal* están al revés las invocaciones a las funciones *cargar()* y *calcular()*: primero debería invocar a *calcular()* y luego a *cargar()*.

d.

Lanza un error de intérprete en la carga de datos: la carga de datos DEBE hacerse en el script principal y no en una función separada.

¡Correcto!

La respuesta correcta es:

El programa lanza un error al ser ejecutado, en el *script principal*: no reconoce las variables *c* y *d* ya que fueron definidas en la función *calcular()*, pero como variables locales a ella.

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

El siguiente programa utiliza variables definidas como globales (ver Ficha 09), ¿Qué hace exactamente el programa (y por qué hace eso)?

```
__author__ = 'Catedra de AED'

def funcion01():
    a = b * 2

def funcion02():
    a = b + c

def test():
    global a, b, c

    a, b, c = 10, 20, 30
    funcion01()
    funcion02()
    print('Valor final de a:', a)

# script principal
test()
```

Seleccione una:

a.

Lanza un error de intérprete al intentar ejecutarlo: una función no puede declarar variables locales con el mismo nombre que una variable definida como global.

b.

Muestra el mensaje: *Valor final de a: 50* - Porque la variable *a* queda valiendo el valor asignado al invocar a *funcion02()*.

c.

Muestra el mensaje: *Valor final de a: 10* - Porque la variable *a* cuyo valor se muestra en *test()*, es *global* y queda valiendo el valor 10 asignado en la misma *test()*. ✓

¡Correcto!

d.

Muestra el mensaje: *Valor final de a: 40* - Porque la variable *a* queda valiendo el valor asignado al invocar a *funcion01()*.

¡Correcto!

La respuesta correcta es:

Muestra el mensaje: *Valor final de a: 10* - Porque la variable *a* cuyo valor se muestra en *test()*, es *global* y queda valiendo el valor 10 asignado en la misma *test()*.

**Pregunta 3**

Correcta

Puntúa 2 sobre 2

Suponga que se desea desarrollar un programa básico en Python que cargue dos números y los muestre ordenados de menor a mayor. El programa que se muestra a continuación tiene ese objetivo. ¿Hay algún problema con el programa mostrado? (tenga cuidado con el concepto de variable local vs. el concepto de variable global -introducidos ambos en la **Ficha 09**).

```
__author__ = 'Cátedra de AED'

def ordenar():
    if a > b:
        men = b
        may = a
    else:
        men = a
        may = b

# script principal...
men, may = 0, 0

a = int(input('Primer número: '))
b = int(input('Segundo número: '))

ordenar ()

print('Menor:', men)
print('Mayor:', may)
```

Seleccione una:

a.

El programa compila y ejecuta, pero muestra resultados incorrectos: se carguen los datos que se carguen, siempre muestra que el mayor y el menor valen 0. ✓

¡Ok! Efectivamente, el problema es que las variables *men* y *may* definidas con valor 0 en el *script principal*, NO SON las mismas *men* y *may* que aparecen definidas localmente en la función *ordenar()*: los valores que se asignan en esta última, se pierden al terminar la función ya que sus variables locales se desalojan de la memoria. El *script principal* no está viendo a esas dos variables, sino a las dos que valen 0, y eso es lo que muestra al final.

b.

El programa lanza un error de intérprete: las variables *men* y *may* se definieron dos veces en dos ámbitos distintos, y eso no puede hacerse en Python.

c.

No hay ningún problema. Muestra los números ordenados correctamente.

d.

El programa lanza un error de intérprete: la función *ordenar()* debió ser declarado debajo del *script principal*.

¡Correcto!

La respuesta correcta es:

El programa compila y ejecuta, pero muestra resultados incorrectos: se carguen los datos que se carguen, siempre muestra que el mayor y el menor valen 0.

**Pregunta 4**

Parcialmente correcta

Puntúa 1 sobre 1

La siguiente función está inspirada en un subproblema de la sección de Temas Avanzados de la Ficha 5, pero de forma que ahora se plantea como una función en la cual todas sus variables se definen como globales (**ver Ficha 09**) y contiene además uno o más errores en su escritura. Identifique cuál o cuáles son esos errores (como puede haber más de uno, entonces más de una respuesta podría ser válida... marque todas las que considere aplicables):

```
def ordenar_dos:  
    global a, b, men, may  
    if a > b:  
        men = b  
        may = a  
    else:  
        men = a  
        may = b
```

Seleccione una o más de una:

a.

Faltan los paréntesis vacíos en la cabecera de la función.

b.

Está mal indentado el bloque de acciones de la función: debería encolumnarse hacia la derecha del inicio de la cabecera. ✓

¡Ok!

c.

La declaración *global* debe ir en la misma línea que la cabecera.

d.

No corresponde colocar el símbolo dos puntos (:) al finalizar la cabecera.

Las respuestas correctas son:

Faltan los paréntesis vacíos en la cabecera de la función.,

Está mal indentado el bloque de acciones de la función: debería encolumnarse hacia la derecha del inicio de la cabecera.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

Considere el problema del cálculo de los montos a pagar por el viaje de estudios de tres cursos de secundaria, tal como se presentó en la Ficha 10. El programa que se muestra más abajo, está replanteado de forma de usar variables globales (**ver Ficha 09**) en lugar de parámetros y retornos, y la función *mayor()* (que determinaba cuál era el curso con más cantidad de alumnos, y además, cuál era el monto a pagar por ese curso), está definida de la siguiente forma:

```

__author__ = 'Cátedra de AED'

def mayor():
    global may, may_cur
    if c1 > c2 and c1 > c3:
        may = m1
        may_cur = 'Primero'
    else:
        if c2 > c3:
            may = m2;
            may_cur = 'Segundo'
        else:
            may = m3
            may_cur = 'Tercero'

def montos():
    global m1, m2, m3
    m1 = c1 * 1360
    m2 = c2 * 1360
    m3 = c3 * 1360

    if c1 > 40:
        m1 = m1 - m1/100*5

    if c2 > 40:
        m2 = m2 - m2/100*5

    if c3 > 40:
        m3 = m3 - m3/100*5

def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = may / mtot * 100
    else:
        porc = 0

def test():
    global c1, c2, c3

    # título general y carga de datos...
    print('Cálculo de los montos de un viaje de estudios...')
    c1 = int(input('Ingrese la cantidad de alumnos del primer curso: '))
    c2 = int(input('Ingrese la cantidad de alumnos del segundo curso: '))
    c3 = int(input('Ingrese la cantidad de alumnos del tercer curso: '))

    # procesos... invocar a las funciones en el orden correcto...
    montos()
    mayor()
    porcentaje()

    # visualización de resultados
    print('El curso mas numeroso es el', may_cur)
    print('El monto del viaje del primer curso es:', m1)
    print('El monto del viaje del segundo curso es:', m2)
    print('El monto del viaje del segundo curso es:', m3)
    print('El porcentaje del monto del mas numeroso en el total es:', porc)

# script principal: sólo invocar a test()...
test()

```

¿Qué efecto causaría en el programa que la función *mayor()* fuese reemplazada por esta otra versión que se muestra a continuación, y qué cambios debería hacer el programador para que su programa siga funcionando y cumpliendo con los mismos requerimientos? (Más de una respuesta puede ser válida. Marque todas las que

considere correctas):

```
def mayor():
    global salida
    if c1 > c2 and c1 > c3:
        salida = 'Primero', m1
    else:
        if c2 > c3:
            salida = 'Segundo', m2
        else:
            salida = 'Tercero', m3
```

Seleccione una o más de una:

a.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[0] / mtot * 100
    else:
        porc = 0
```

b.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:

```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[0])
print('El monto del viaje del primer curso es:', m1)
print('El monto del viaje del segundo curso es:', m2)
print('El monto del viaje del tercero curso es:', m3)
print('El porcentaje del monto del mas numeroso en el total es:', porc)
```

✓ ¡Ok! Efectivamente, en la tupla *salida*, la descripción del curso mayor está en la posición 0, y el monto mayor está en la 1.

c.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[1] / mtot * 100
    else:
        porc = 0
```

✓ ¡Ok! Efectivamente, en la tupla *salida*, el monto mayor está en la posición 1, mientras que la descripción del curso mayor, está en la cero.

d.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:

```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[1])
print('El monto del viaje del primer curso es:', m1)
print('El monto del viaje del segundo curso es:', m2)
print('El monto del viaje del segundo curso es:', m3)
print('El porcentaje del monto del mas numeroso en el total es:', porc)
```

¡Correcto!

Las respuestas correctas son:

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[1] / mtot * 100
    else:
        porc = 0
```

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:

```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[0])
print('El monto del viaje del primer curso es:', m1)
print('El monto del viaje del segundo curso es:', m2)
print('El monto del viaje del segundo curso es:', m3)
print('El porcentaje del monto del mas numeroso en el total es:', porc)
```

**Pregunta 6**

Parcialmente  
correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son CIERTAS respecto del concepto de *reutilización de piezas de software* en programación? [Aclaración: varias respuestas pueden ser válidas. Marque todas las que considere correctas]

Seleccione una o más de una:

a.

La reutilización es el proceso por el cual un mismo segmento de código fuente se copia y se pega en cuanto lugar el programador necesite volver a contar con dicho segmento.

b.

La reutilización es el proceso por el cual una misma función o módulo ya desarrollado para un programa, vuelve a usarse en otro programa en forma directa, por simple mecanismo de invocación y sin tener que modificar el nuevo programa o la propia función/módulo.

c.

Los lenguajes de programación proveen grandes cantidades de funciones reutilizables en sus librerías de funciones predefinidas o estándar. ✓

¡Ok!

d.

La reutilización permite hacer el trabajo en forma más eficiente, ordenada y profesional a un programador.



✓

¡Ok!

Las respuestas correctas son:

La reutilización es el proceso por el cual una misma función o módulo ya desarrollado para un programa, vuelve a usarse en otro programa en forma directa, por simple mecanismo de invocación y sin tener que modificar el nuevo programa o la propia función/módulo.,

Los lenguajes de programación proveen grandes cantidades de funciones reutilizables en sus librerías de funciones predefinidas o estándar.,

La reutilización permite hacer el trabajo en forma más eficiente, ordenada y profesional a un programador.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

En la columna de la izquierda se muestran diversos planteos posibles para una función que debe calcular el promedio de todos los números enteros contenidos en el intervalo [1, n]. Algunas (o todas) presentan inconvenientes en cuanto sus posibilidades de reutilización. Para cada versión, seleccione desde la lista de la columna derecha la opción que mejor describe su problema o planteo:



Acoplamiento procesos/interfaz respecto de los datos y los resultados que debe gestionar la función ▾



Acoplamiento procesos/interfaz respecto de los datos que debe procesar la función ▾



Dependencia externa en relación a los resultados generados por la función ▾



Planteo genérico y reutilizable ▾



¡Correcto!

```
def promedio():
```

```
    n = int(input('N: '))
    ac = 0
    for x in range(1, n+1):
        ac += x
    p = ac / n
    print('Promedio:', p)
```

```
def promedio(n):
```

```
    global p
    ac = 0
    for x in range(1, n+1):
        ac += x
    p = ac / n
```

```
    def promedio(n):
```

```
        ac = 0
        for x in range(1, n+1):
            ac += x
        p = ac / n
        return p
```

La respuesta correcta es: → Acoplamiento procesos/interfaz respecto de los datos y los resultados que debe gestionar la función, → Acoplamiento procesos/interfaz respecto de los datos que debe procesar la función,

```
def promedio():
    n = int(input('N: '))
    ac = 0
    for x in range(1, n+1):
        ac += x
    p = ac / n
    return p
```

→ Dependencia externa en relación a los resultados generados por la

función,

→ Planteo genérico y reutilizable

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son verdaderas respecto de la relación entre *Programación Estructurada* y *Programación Modular*? (Más de una respuesta puede ser cierta. Marque todas las que considere correctas)

Seleccione una o más de una:

- a.

La Programación Estructurada y la Programación Modular son dos paradigmas de programación diferentes, sin conexión entre ellos.

- b.

Los conceptos de Programación Estructurada y Programación Modular son equivalentes.

- c.

La Programación Modular enuncia a la Programación Estructurada como uno de sus principios de trabajo, pero no se limita a ella.

- d.

La Programación Estructurada enuncia a la Programación Modular como uno de sus principios de trabajo, pero no se limita a ella. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

La Programación Estructurada enuncia a la Programación Modular como uno de sus principios de trabajo, pero no se limita a ella.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Cuáles de los siguientes elementos favorecen la *reutilización de una función*? [Aclaración: varias respuestas pueden ser válidas. Marque todas las que considere correctas]

Seleccione una o más de una:

- a.

La correcta escritura de la función apoyándose a principios, reglas y consejos de buenas prácticas (usar nombres descriptivos, emplear comentarios, dejar espacios en blanco donde contribuya a una mejor lectura, etc).

- b.

La declaración de la función de forma que acepte parámetros para gestionar sus datos, y emplee el mecanismo de retorno para devolver sus resultados. ✓

¡Ok!

- c.

La independencia de la función en relación a elementos definidos fuera de ella. ✓

¡Ok!

- d.

La independencia de la función en referencia a procesos de carga de datos y visualización de resultados.



¡Ok!

¡Correcto!

Las respuestas correctas son:

La independencia de la función en relación a elementos definidos fuera de ella.,

La independencia de la función en referencia a procesos de carga de datos y visualización de resultados.,

La declaración de la función de forma que acepte parámetros para gestionar sus datos, y emplee el mecanismo de retorno para devolver sus resultados.

**Pregunta 10**

Correcta

Puntúa 1 sobre 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 [1, n]?

Seleccione una:

a.

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

b.

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

c.

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

d.

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

¡Ok! Efectivamente, si un número *i* es par (o sea, divisible por 2) y es divisible por 3 y por 7 al mismo tiempo, entonces *i* es de la forma  $i = 2 * 3 * 7 * k = 42 * k$  (con  $k = 0, 1, 2, \dots$ ) Por lo tanto, el número *i* es también múltiplo de 42. Un range que comience en 0, y luego incluya números de 42 en 42, contendrá los números pedidos (el propio 0 es divisible por 42...)

¡Correcto!

La respuesta correcta es:

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

**Pregunta 11**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes funciones creará una tupla conteniendo exclusivamente los números divisibles por 4 y por 5 al mismo tiempo, del intervalo [1, n]?

Seleccione una:

a.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 5):  
        r += i,  
    return r
```

b.

```
def prueba(n):  
    r = ()  
    for i in range(4, n+1, 5):  
        r += i,  
    return r
```

c.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 4):  
        r += i,  
    return r
```

d.

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 20):  
        r += i,  
    return r ✓
```

¡Ok! Efectivamente, si un número  $i$  es divisible por 4 y por 5 al mismo tiempo, entonces  $i$  es de la forma  $i = 4 * 5 * k = 20 * k$  (con  $k = 0, 1, 2, \dots$ ) Por lo tanto, el número  $i$  es también múltiplo de 20. Un range que comience en 0, y luego incluya números de 20 en 20, contendrá los números pedidos (el propio 0 es divisible por 20...)

¡Correcto!

La respuesta correcta es:

```
def prueba(n):  
    r = ()  
    for i in range(0, n+1, 20):  
        r += i,  
    return r
```

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

¿Cuál es el motivo por el cual el *acoplamiento entre procesos e interfaz de usuario* es un problema en cuanto a las posibilidades de reuso de una función? [Más de una opción puede ser válida... marque **todas** las que considere correctas]

Seleccione una o más de una:

 a.

El concepto de acoplamiento entre procesos e interfaz de usuario no es aplicable a funciones, sino a diagramas de flujo y pseudocódigos. Por lo tanto, la pregunta en sí misma carece de sentido.

 b.

Si la función procede a ingresar desde la interfaz de usuario el valor de alguna variable, pero el programador no necesitaba esa carga debido a que sus variables venían con valores definidos previamente, entonces deberá modificar la función para adaptarla al funcionamiento de su programa. Tendrá un problema similar si la función despliega en pantalla algún resultado. ✓

¡Ok!

 c.

En la medida de lo posible, el acoplamiento entre procesos e interfaz de usuario debería ser evitado, pero en algún momento el programador necesitará funciones que hagan el trabajo de gestionar la interfaz con el usuario. Tendrá entonces ciertas funciones con objetivos de control de interfaz, y otras funciones genéricas y reutilizables. ✓

¡Ok!

 d.

Si en la función se supone que la interfaz de usuario estará soportada en la consola estándar, y luego se pretendiese reusarla pero en un contexto de ventanas y componentes visuales de alto nivel, entonces la función deberá ser rediseñada completamente. ✓

¡Ok!

¡Correcto!

Las respuestas correctas son:

Si la función procede a ingresar desde la interfaz de usuario el valor de alguna variable, pero el programador no necesitaba esa carga debido a que sus variables venían con valores definidos previamente, entonces deberá modificar la función para adaptarla al funcionamiento de su programa. Tendrá un problema similar si la función despliega en pantalla algún resultado.,

En la medida de lo posible, el acoplamiento entre procesos e interfaz de usuario debería ser evitado, pero en algún momento el programador necesitará funciones que hagan el trabajo de gestionar la interfaz con el usuario. Tendrá entonces ciertas funciones con objetivos de control de interfaz, y otras funciones genéricas y reutilizables.,

Si en la función se supone que la interfaz de usuario estará soportada en la consola estándar, y luego se pretendiese reusarla pero en un contexto de ventanas y componentes visuales de alto nivel, entonces la función deberá ser rediseñada completamente.

**Pregunta 13**

Correcta

Puntúa 1 sobre 1

¿Cuál es el motivo por el cual la *dependencia externa* es un problema en cuanto a las posibilidades de reuso de una función?

Seleccione una:

a.

Es peor que eso: si la función utiliza elementos definidos fuera de ella, no hay forma alguna de poder usarla en ningún programa: no sólo no es reutilizable, sino que simplemente no puede usarse.

b.

El concepto de dependencia externa no es aplicable a funciones, sino a ciclos. Por lo tanto, la pregunta en sí misma carece de sentido.

c.

Una función puede tener dependencia externa, pero esa dependencia externa no es un problema en cuanto a la reutilización de una función.

d.

Si la función utiliza elementos definidos fuera de ella, entonces los programas donde se quiera usar esa función deberán hacer declaraciones especiales para poder usarla, y/o se deberá modificar la propia función ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

Si la función utiliza elementos definidos fuera de ella, entonces los programas donde se quiera usar esa función deberán hacer declaraciones especiales para poder usarla, y/o se deberá modificar la propia función

**Pregunta 14**

Correcta

Puntúa 3 sobre 3

En la columna de la izquierda se muestran diversas situaciones que requieren algún tipo de análisis y cálculo combinatorio. Seleccione de la lista de la columna de la derecha el tipo de cálculo que correspondería aplicar para cada caso.

Calcular cuántas variantes pueden formarse con las letras y los dígitos de la patente de un automóvil en Argentina ✓  
(tres letras del alfabeto español, sin la ñ, 26 letras en total) y tres dígitos al final.

Principio básico de conteo  $[t = e1 * e2 * e3 * \dots * ek]$

Calcular cuántos grupos diferentes de  $m = 4$  colores, pueden formarse a partir de un conjunto de  $n = 7$  colores, admitiendo colores repetidos.

Combinaciones de  $n$  elementos tomados de  $a m$ , con reposición  $[cr(n, m) = (m + (n - 1))! / (m! * (n - 1)!)]$

Calcular en cuántas formas podrían organizarse equipos de  $m = 4$  participantes de un evento de búsqueda del tesoro, sabiendo que hay  $n = 28$  inscriptos en total.

Combinaciones de  $n$  elementos tomados de  $a m$ , sin reposición  $[c(n, m) = n! / (m! * (n - m)!)]$

Calcular cuántos arreglos diferentes de  $m = 3$  letras se pueden formar con las  $n = 6$  letras de la palabra 'Python'.

Permutaciones de  $n$  elementos tomados de  $a m$ , con reposición  $[pr(n, m) = pow(n, m)]$

Calcular en cuántas formas podría quedar organizado el orden de mérito final de los m = 5 cargos a cubrir de un concurso docente en el que participan n = 12 posibles candidatos.

▼ Permutaciones de n elementos tomados de a m, sin reposición [ $p(n, m) = n! / (n - m)!$ ]

¡Correcto!

La respuesta correcta es:

Calcular cuántas variantes pueden formarse con las letras y los dígitos de la patente de un automóvil en Argentina (tres letras del alfabeto español, sin la ñ, 26 letras en total) y tres dígitos al final. → Principio básico de conteo [ $t = e1 * e2 * e3 * ... * ek$ ],

Calcular cuántos grupos diferentes de m = 4 colores, pueden formarse a partir de un conjunto de n = 7 colores, admitiendo colores repetidos. → Combinaciones de n elementos tomados de a m, con reposición [ $cr(n, m) = (m + (n - 1))! / (m! * (n - 1)!)$ ],

Calcular en cuántas formas podrían organizarse equipos de m = 4 participantes de un evento de búsqueda del tesoro, sabiendo que hay n = 28 inscriptos en total. → Combinaciones de n elementos tomados de a m, sin reposición [ $c(n, m) = n! / (m! * (n - m)!)$ ],

Calcular cuántos arreglos diferentes de m = 3 letras se pueden formar con las n = 6 letras de la palabra 'Python'. → Permutaciones de n elementos tomados de a m, con reposición [ $pr(n, m) = pow(n, m)$ ],

Calcular en cuántas formas podría quedar organizado el orden de mérito final de los m = 5 cargos a cubrir de un concurso docente en el que participan n = 12 posibles candidatos. → Permutaciones de n elementos tomados de a m, sin reposición [ $p(n, m) = n! / (n - m)!$ ]

---

**Comenzado el** sábado, 16 de junio de 2018, 17:59

**Estado** Finalizado

---

**Finalizado en** sábado, 16 de junio de 2018, 18:38

---

**Tiempo empleado** 39 minutos 44 segundos

---

**Puntos** 13/13

---

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Suponga la siguiente cabecera de una función cuyo bloque de acciones es irrelevante a los efectos de la pregunta:

```
def ejemplo(a, b=0, c='Desconocido', d='Ingeniero'):  
    # el bloque de acciones no importa ahora...
```

¿Cuáles de las siguientes invocaciones a esta función son correctas y activan la función sin producir un error de intérprete? (Observación: más de una respuesta puede ser correcta. Marque todas las que considere válidas... y tómese su tiempo para pensar y probar cada respuesta posible...)

Seleccione una o más de una:



a.

`ejemplo('Medico')` ✓

Correcto. Efectivamente, un solo parámetro es obligatorio en esta función y por lo tanto el único valor enviado ('Medico') se asigna en ese parámetro (la variable *a*). El resto toma sus valores por default [Valores finales de los parámetros: *a = Medico, b = 0, c = "Desconocido", d = "Ingeniero"*] Si este resultado le parece extraño, ¡recuerde que los parámetros se toman y se asignan por orden de aparición, y que Python es un lenguaje de tipado dinámico!]



b.

`ejemplo()`

c.

`ejemplo(32, 20)` ✓

Correcto. Efectivamente, el primer parámetro es obligatorio (y en este caso es un número), y el segundo es opcional dado que admite un valor default. Los otros dos quedan valiendo valores default [Valores finales de los parámetros: *a = 32, b = 20, c = "Desconocido", d = "Ingeniero"*]



d.

`ejemplo(40, 'Juan', 'Abogado')` ✓

Correcto. Efectivamente, el primer parámetro es obligatorio (y en este caso es un número). El segundo y el tercero quedan valiendo las cadenas "Juan" y "Abogado". Y el restante queda valiendo su valor default [Valores finales de los parámetros: *a = 40, b = "Juan", c = "Abogado", d = "Ingeniero"*] Si este resultado le parece extraño, ¡recuerde que los parámetros se toman y se asignan por orden de aparición, y que Python es un lenguaje de tipado dinámico!]

¡Correcto!

Las respuestas correctas son:

`ejemplo(32, 20),``ejemplo(40, 'Juan', 'Abogado'),`

```
ejemplo('Medico')
```

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

Suponga que se quiere desarrollar una función que tome como parámetro una secuencia (un string, una lista, una tupla, o cualquier otro tipo de colección que permita acceso mediante índices en Python) y que proceda a ordenar esa colección con diversos algoritmos conocidos, en forma ascendente o descendente a elección de quien invoca a la función.

En ese contexto, considere la siguiente definición para esa función, en la cual no es relevante su bloque de acciones a los efectos de la pregunta:

```
def ordenar(lista, ascendente=True,  
          algoritmo='Quicksort'):  
    # el bloque de acciones no importa ahora...
```

¿Cuáles de las siguientes invocaciones a esta función son correctas, y no provocarán un error de intérprete? (Observación: más de una respuesta puede ser válida. Marque todas las que considere adecuadas... y de nuevo: tómese su tiempo!!!)

Seleccione una o más de una:

 a.

```
ordenar('azbycx', algoritmo='Shellsort')
```

 ✓

Correcto. Efectivamente, el primer parámetro es obligatorio en esta función. El segundo queda con valor default. Y el tercero es seleccionado por palabra clave [Valores finales de los parámetros: lista = "azbycx", ascendente = True, algoritmo = "Shellsort"]

 b.

```
ordenar('abcdef', ascendente=False, 'Insertionsort')
```

 c.

```
ordenar('abcdef', False, metodo='Bubblesort')
```

 d.

```
ordenar(lista=(1,2,3),  
        algoritmo='Heapsort',  
        ascendente=False)
```

 ✓

Correcto. Efectivamente, el primer parámetro es obligatorio en esta función, y no sólo eso sino que en este caso se lo está accediendo por palabra clave y queda valiendo una tupla compuesta de números. El segundo y el tercero son seleccionados por palabra clave, sin importar si se los está accediendo en orden diferente al de su declaración. [Valores finales de los parámetros: lista = (1,2,3), ascendente = False, algoritmo = "Heapsort"]

¡Correcto!

Las respuestas correctas son:

```
ordenar('azbycx', algoritmo='Shellsort'),
```

```
ordenar(lista=(1,2,3), algoritmo='Heapsort', ascendente=False)
```

**Pregunta 3**

Correcta

Puntúa 2 sobre 2

La siguiente función se propone tomar como parámetro el nombre de un empleado y los importes de los últimos sueldos que percibió para luego mostrar un listado que incluya el nombre recibido y el promedio de sus sueldos:

```
def procesar(nombre, *importes):
    print('Nombre del empleado:', nombre)

    p = 0
    n = len(importes)
    if n != 0:
        s = 0
        for imp in importes:
            s += imp
        p = s / n
    print('Sueldo promedio:', p)
```

¿Cuáles de las siguientes invocaciones son correctas, en el sentido de ejecutarse sin problemas y mostrar los resultados pedidos sin provocar una interrupción del programa? (Observación: más de una respuesta puede ser válida, por lo que marque todas las que considere oportunas).

Seleccione una o más de una:



a.

```
procesar('Laura', 1000, '2000', True)
```



b.

```
procesar('Carlos', '1000', '2000', '3000')
```



c.

```
procesar('Pedro', 1234.56, 2345.45) ✓
```

Correcto. La tupla **importes** tiene dos valores numéricos de tipo *float*, que son procesados correctamente, entregando un sueldo promedio de 1208.1866666666667



d.

```
procesar('Luis') ✓
```

Correcto. En este caso, la tupla **importes** está vacía, y se muestra un sueldo promedio igual a cero, correctamente.

¡Correcto!

Las respuestas correctas son:

```
procesar('Luis'),
procesar('Pedro', 1234.56, 2345.45)
```

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **incorrecta** en relación al concepto de **módulo** en Python, y/o a elementos asociados al uso de módulos en Python?

Seleccione una:

a.

Un módulo pensado para *ser ejecutado en forma directa* debería contener un script de control de la forma `if __name__ == '__main__'` para evitar que al ser importado se ejecute en forma inconveniente su posible script principal.

b.

Cuando se usa una instrucción `import`, Python busca primero si existe un módulo estándar cuyo nombre coincide con el del módulo importado. Si no lo encuentra, busca entonces en la lista de carpetas indicada por la variable `sys.path`.

c.

Un módulo en Python es cualquier archivo con extensión `.py` (código fuente que contenga funciones, definiciones generales, clases, variables, etc.)

d.

Un módulo siempre puede ser importado desde otro módulo, mediante instrucciones `import` o `from import`.

e.

Un módulo en Python puede tener elementos `docstring` que documenten su uso y su contenido.

f.

Cada vez que un módulo es importado en un programa, su contenido es vuelto a compilar por el intérprete. ✓

¡Correcto! Esta es justamente la afirmación incorrecta pedida: no es cierto que un módulo se compila nuevamente cada vez que se lo importa. Sólo se compila la primera vez, y esa precompilación se almacena en la carpeta `__pycache__` del proyecto. Con esto se gana tiempo de ejecución, al no tener que compilar una y otra vez un módulo cada vez que un programa lo importa o lo accede.

¡Correcto!

La respuesta correcta es:

Cada vez que un módulo es importado en un programa, su contenido es vuelto a compilar por el intérprete.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Suponga las siguientes instrucciones *import* en un programa (note que todos los módulos nombrados en esos *import* pertenecen a la librería estándar de Python):

```
import math
from re import split
from random import random
```

¿Cuáles de las siguientes invocaciones a funciones son **correctas**, considerando los *import* que se acaban de mostrar? (Observación: más de una respuesta puede ser válida. Marque todas las que considere oportunas)

Seleccione una o más de una:

a.

```
# observación: la función split() (partir en subcadenas)
pertenece al modulo re
y = split("\L+", "La ola de la vida") ✓
```

Correcto. Hay una instrucción de importación que específicamente introduce el nombre de la función *split()* en este script, por lo cual no debe usarse el prefijo "*re*."

b.

```
# observación: la función random() pertenece al modulo
random
z = random.random()
```

c.

```
# observación: la función sqrt() (raíz cuadrada) pertenece
al modulo math
x = math.sqrt(4) ✓
```

Correcto. Hay un import que introduce el nombre del módulo *math* en este programa, y a partir de allí cada función de ese módulo debe ser nombrada con el prefijo "*math*."

d.

```
# observación: la función log2() (logaritmo en base 2)
pertenece al modulo math
lg = log2(8)
```

¡Correcto!

Las respuestas correctas son:

```
# observación: la función sqrt() (raíz cuadrada) pertenece al
modulo math
x = math.sqrt(4),

# observación: la función split() (partir en subcadenas)
pertenece al modulo re
y = split("\L+", "La ola de la vida")
```

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

En la columna de la izquierda, se nombran algunos de los módulos que existen en la librería estándar de Python. Seleccione el uso o aplicación de cada uno de estos módulos.

sys	Variables de entorno y acceso a parámetros de línea de órdenes ▾
doctest	Validaciones de tests incluidos en strings de documentación. ▾
datetime	Manipulación de fechas y horas ▾
urllib.request	Recuperación de datos desde un url. ▾
re	Reconocimiento de patrones ▾
xml.dom - xml.sax	Parsing de documentos XML. ▾
os	Acceso a funciones del sistema operativo. ▾

¡Correcto!

La respuesta correcta es: sys → Variables de entorno y acceso a parámetros de línea de órdenes, doctest → Validaciones de tests incluidos en strings de documentación., datetime → Manipulación de fechas y horas, urllib.request → Recuperación de datos desde un url., re → Reconocimiento de patrones, xml.dom - xml.sax → Parsing de documentos XML., os → Acceso a funciones del sistema operativo.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

Suponga la definición del siguiente módulo en Python, **tal como se muestra**, y suponga también que este módulo se ha almacenado en el archivo "cadenas.py":

```
def mensaje(m):  
    print('El mensaje es:', m)  
  
def invertir(m):  
    print('El mensaje (invertido) es:')  
  
    n = len(m)  
    for i in range(n-1, -1, -1):  
        print(m[i], end='')  
  
def test():  
    cadena = input('Ingrese un mensaje: ')  
    mensaje(cadena)  
    invertir(cadena)  
  
test()
```

Suponga ahora que se define otro módulo (almacenado en el archivo "prueba.py" y en la misma carpeta que el módulo "cadenas.py") cuyo contenido es el siguiente, **tal como se muestra**:

```
import cadenas  
  
def principal():  
    cad1 = 'Universidad Tecnologica Nacional'  
    cadenas.mensaje(cad1)  
    cadenas.invertir(cad1)  
  
if __name__ == '__main__':  
    principal()
```

¿Cuál de las siguientes afirmaciones describe correctamente lo que ocurrirá si se pide ejecutar el contenido del módulo "prueba.py"?

Seleccione una:

a.

El módulo prueba.py está importando al módulo cadenas.py y como este último **no** incluye un chequeo de la forma `if __name__ == "__main__"`, entonces al ejecutar prueba.py se ejecutará primero la función prueba.principal() y luego la función cadenas.test().

b.

El módulo prueba.py está importando al módulo cadenas.py y como este último **no** incluye un chequeo de la forma `if __name__ == "__main__"`, entonces al ejecutar prueba.py se producirá un error de intérprete cuando se intente cargar el módulo cadenas.py.

c.

El módulo prueba.py está importando al módulo cadenas.py y como prueba.py incluye un chequeo de la forma `if __name__ == '__main__'`, entonces al ejecutar prueba.py se ejecutará solamente la función prueba.principal().

d.

El módulo prueba.py está importando al módulo cadenas.py y como este último **no** incluye un chequeo de la forma `if __name__ == '__main__'`, entonces al ejecutar prueba.py se ejecutará primero la función cadenas.test() y luego la función prueba.principal(). ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

El módulo prueba.py está importando al módulo cadenas.py y como este último **no** incluye un chequeo de la forma `if __name__ == '__main__'`, entonces al ejecutar prueba.py se ejecutará primero la función cadenas.test() y luego la función prueba.principal().

#### Pregunta 8

Correcta

Puntúa 1 sobre 1

Para cada una de las variables predefinidas de uso global de Python de la primera columna, seleccione la definición que le corresponde o que mejor se le ajuste.

- |                         |  |
|-------------------------|--|
| <code>__author__</code> | El nombre del autor del elemento. ▾ ✓                                    |
| <code>__name__</code>   | El nombre de un módulo o el valor literal ' <code>__main__</code> '. ▾ ✓ |
| <code>__all__</code>    | Los elementos a importar desde un paquete. ▾ ✓                           |
| <code>__doc__</code>    | Todos los docstrings que contiene el elemento. ▾ ✓                       |

¡Correcto!

La respuesta correcta es: `__author__` → El nombre del autor del elemento., `__name__` → El nombre de un módulo o el valor literal '`__main__`', `__all__` → Los elementos a importar desde un paquete., `__doc__` → Todos los docstrings que contiene el elemento.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

Suponga que se tiene un *paquete* en Python llamado **soporte**, y que ese paquete contiene cuatro módulos llamados respectivamente *modelo*, *persistencia*, *interfaz* y *excepciones*. ¿Cuáles de las siguientes **son correctas** en relación al archivo `__init__.py` del paquete **soporte**? (Más de una puede ser válida... marque todas las que considere apropiadas)

Seleccione una o más de una:

 a.

El archivo `__init__.py` del paquete **soporte** no puede dejarse vacío. Debe asignar los nombres de los módulos que posee, en la variable `__all__`.

 b.

El archivo `__init__.py` del paquete **soporte** puede dejarse vacío. No importa si el paquete tiene subpaquetes y módulos o no. ✓

Correcto... dejar vacío el archivo `__init__.py` es válido... aunque a veces podría no tener mucha utilidad.

 c.

El archivo `__init__.py` del paquete **soporte** debe estar presente en la carpeta de ese paquete, para que Python lo reconozca como un paquete válido. ✓

Correcto... justamente, la presencia de ese archivo es lo que hace que Python considere a la carpeta como un package.

 d.

El archivo `__init__.py` del paquete **soporte** *puede* contener una asignación con los nombres de todos o algunos de los módulos que posee, en la variable `__all__`. ✓

Correcto... sabemos que se puede usar la variable `__all__` para asignar en ella los nombres de los subpaquetes y/o módulos que queremos que se importen si se hace un **from soporte import \***.

¡Correcto!

Las respuestas correctas son:

El archivo `__init__.py` del paquete **soporte** puede dejarse vacío. No importa si el paquete tiene subpaquetes y módulos o no.,

El archivo `__init__.py` del paquete **soporte** *puede* contener una asignación con los nombres de todos o algunos de los módulos que posee, en la variable `__all__`.,

El archivo `__init__.py` del paquete **soporte** debe estar presente en la carpeta de ese paquete, para que Python lo reconozca como un paquete válido.

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

Suponga que se tiene un paquete llamado `test` en Python, y que ese paquete contiene tres subpaquetes llamados `prueba1`, `prueba2` y `prueba3`. Suponga además que archivo `__init__.py` del paquete `test` contiene solo el siguiente `docstring` (y ninguna otra asignación o script adicional):

```
"""El paquete contiene subpaquetes para operaciones de
testing de un sistema.

Lista de subpaquetes incluidos:
:prueba1: Contiene un modulo con funciones para testear
operaciones de input/output
:prueba2: Contiene dos modulos con funciones para
testear manejo de excepciones
"""

¿Hay algún problema con este bloque, en relación al respeto de las convenciones de documentación docstring? (Marque la respuesta que mejor describa la situación)
```

Seleccione una:

a.

Hay un par de problemas: debería haber una línea en blanco después de la primera línea del bloque (la descripción de los subpaquetes debería aparecer luego de esa línea en blanco); y además, faltaría la descripción del subpaquete `prueba3`. ✓

Correcto. Las convenciones generales sugieren la línea en blanco faltante, y las convenciones para describir un paquete sugieren que se documente brevemente todo subpaquete que el paquete contenga (salvo que algún subpaquete se haya excluido por no haberlo asignado en la variable `__all__`... ¡que no es el caso!)

b.

No hay ningún problema.

c.

Hay un solo y único problema: debería haber una línea en blanco después de la primera línea del bloque (la descripción de los subpaquetes debería aparecer luego de esa línea en blanco)

d.

El problema es el propio bloque en sí mismo: un paquete no lleva documentación `docstring`.

¡Correcto!

La respuesta correcta es:

Hay un par de problemas: debería haber una línea en blanco después de la primera línea del bloque (la descripción de los subpaquetes debería aparecer luego de esa línea en blanco); y además, faltaría la descripción del subpaquete `prueba3`.

**Comenzado el** domingo, 29 de julio de 2018, 11:01

**Estado** Finalizado

**Finalizado en** lunes, 30 de julio de 2018, 06:59

**Tiempo empleado** 19 horas 58 minutos

**Puntos** 14/14

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es CIERTA respecto del uso de índices en un arreglo (representado con una variable de tipo *list*) en Python? Más de una respuesta puede ser válida. Marque todas las que considere correctas.

Seleccione una o más de una:

a.

Los índices de un arreglo van numerados de 0 hasta el tamaño del arreglo menos uno. ✓

¡Correcto!

b.

Los índices de un arreglo deben ser numéricos, y valen como índices tanto valores negativos, como cero o positivos. ✓

¡Correcto!

c.

En Python, el programador puede hacer que los índices de un arreglo comiencen desde un número mayor a cero.

d.

Se pueden usar índices de tipo *cadena de caracteres* para entrar a una casilla de un arreglo.

¡Correcto!

Las respuestas correctas son:

Los índices de un arreglo deben ser numéricos, y valen como índices tanto valores negativos, como cero o positivos.,

Los índices de un arreglo van numerados de 0 hasta el tamaño del arreglo menos uno.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Qué hace el siguiente script en Python?

```
n = 20  
v = n * [0]  
for i in range(n):  
    v[i] = 3*i
```

Seleccione una:

 a.

Crea un arreglo unidimensional de 20 componentes, y llena ese arreglo con los números del 0 al 19.

 b.

Crea un arreglo unidimensional de 20 componentes, y llena ese arreglo con los primeros 20 múltiplos de 3. ✓

¡Correcto!

 c.

Crea un arreglo unidimensional de 20 componentes, y llena ese arreglo con 20 números aleatorios.

 d.

Crea un arreglo unidimensional de 20 componentes, y llena ese arreglo con el número 3 (veinte veces el 3).

¡Correcto!

La respuesta correcta es:

Crea un arreglo unidimensional de 20 componentes, y llena ese arreglo con los primeros 20 múltiplos de 3.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es CIERTA respecto del uso arreglos (representados con variables de tipo *list*) en Python? Más de una respuesta puede ser válida. Marque todas las que considere correctas.

Seleccione una o más de una:

a.

El primer índice de cada dimensión de un arreglo en Python es siempre cero (a menos que se usen índices negativos).

¡Correcto!

b.

Los arreglos implementados como variables de tipo *list* en Python pueden aumentar o disminuir su tamaño a medida que el programador lo requiera.

¡Correcto!

c.

Un arreglo implementado como una variable de tipo *list* es completamente equivalente a una *tuple* en Python.

d.

Siempre se puede saber el tamaño de un arreglo llamando a la función *len()* provista por Python.

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Los arreglos implementados como variables de tipo *list* en Python pueden aumentar o disminuir su tamaño a medida que el programador lo requiera.,

Siempre se puede saber el tamaño de un arreglo llamando a la función *len()* provista por Python.,

El primer índice de cada dimensión de un arreglo en Python es siempre cero (a menos que se usen índices negativos).

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Analice el siguiente script en Python:

```
n = 6
v = n * [0]
for i in range(n):
    v[i] = '123'
```

¿Es correcto este script, o existe algún problema con él?

Seleccione una:

 a.

Lanza un error de índice fuera de rango y se interrumpe, pues se intenta acceder a una casilla con índice fuera de rango en el ciclo for.

 b.

El arreglo fue creado como un arreglo de  $n$  valores de tipo *int*, y por lo tanto no pueden asignarse luego cadenas de caracteres en sus casilleros.

 c.

No hay nada de malo con ese segmento. ✓

¡Correcto! Efectivamente, el arreglo comienza con seis ceros, y luego se cambian esos seis ceros por la cadena '123' repetida seis veces... Y esto es válido por ser Python un lenguaje de tipado dinámico.

 d.

El arreglo *v* está mal definido: deben usarse paréntesis ( o sea:  $v = n * (0)$ ) en lugar de corchetes (o sea, en lugar de:  $v = n * [0]$ )

¡Correcto!

La respuesta correcta es:

No hay nada de malo con ese segmento.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Analice el siguiente script en Python:

```
n = 6
v = n * [0]
for i in range(n+1):
    v[i] = i
```

¿Hay algún problema con el script mostrado?

Seleccione una:

a.

No hay nada de malo con ese script.

b.

El arreglo está mal definido: la instrucción `v = n * [0]` está creando un arreglo vacío, sin casilleros.

c.

Lanza un error y se interrumpe, pues en el ciclo `for` se intenta acceder a una casilla no definida. ✓

¡Correcto! En efecto, el ciclo `for` está incluyendo una vuelta con `i = 6...` y el índice de la última casilla es `5...`

d.

Convierte cada casilla del vector a una cadena de caracteres.

¡Correcto!

La respuesta correcta es:

Lanza un error y se interrumpe, pues en el ciclo `for` se intenta acceder a una casilla no definida.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

Analice el siguiente script en Python:

```
v = [2, 4, 1, 6]
v[0] = v[v[0]] * 3
print('v[0]:', v[0])
```

¿Cuál de las siguientes es correcta en relación al script mostrado?

Seleccione una:

 a.Se mostrará el mensaje:  $v[0] = 12$ . b.Se mostrará el mensaje:  $v[0] = 2$ . c.

Se lanzará un error y el script se interrumpirá.

 d.Se mostrará el mensaje:  $v[0] = 3$ . 

¡Correcto!

¡Correcto!

La respuesta correcta es:

Se mostrará el mensaje:  $v[0] = 3$ .

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Analice el siguiente script en Python:

```
n = 5
a = n * [0]
for i in range(n):
    a[i] = i + 1

v = n * [0]
for i in range(n):
    v[a[i]] = a[i]
```

¿Cuál de las siguientes es correcta en relación al script mostrado?

Seleccione una:

a.

El arreglo *v* queda valiendo los mismos valores que el arreglo *a*, pero convertidos al tipo *float*.

b.

Tanto el arreglo *a* como el *v* quedan con todos sus casilleros valiendo 0.

c.

El script lanza un error y se interrumpe por intentar acceder a una casilla fuera de rango en el arreglo *v*. ✓

¡Correcto! Los valores de los casilleros de *a* se están usando como índices para entrar en *v*... pero el último casillero de *a* está valiendo 5, con lo cual se produce un error en la última vuelta del segundo *for*.

d.

Todos los casillas del arreglo *a* se convierten a *float* y se vuelven a asignar en el mismo arreglo *a*.

¡Correcto!

La respuesta correcta es:

El script lanza un error y se interrumpe por intentar acceder a una casilla fuera de rango en el arreglo *v*.

**Pregunta 8**

Correcta

Puntúa 2 sobre 2

¿Qué hace el siguiente programa en Python?

```
__author__ = 'Cátedra de AED'

def test():
    n = 10
    v = n * [0]

    for i in range(n):
        v[i] = int(input('v[' + str(i) + ']: '))

    im = 0
    for i in range(1, n):
        if v[i] < v[im]:
            im = i

    print('El valor pedido es:', v[im])

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

Seleccione una:

 a.

Carga por teclado un arreglo unidimensional con 10 números enteros. Luego busca el mayor valor contenido en el arreglo y muestra ese mayor.

 b.

Carga por teclado un arreglo unidimensional con 10 números enteros. Luego calcula y muestra el promedio de los elementos contenidos en ese arreglo.

 c.

Carga por teclado un arreglo con 10 números enteros y luego ordena y muestra ese arreglo.

 d.

Carga por teclado un arreglo unidimensional con 10 números enteros. Luego busca el menor valor contenido en el arreglo y muestra ese menor. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Carga por teclado un arreglo unidimensional con 10 números enteros. Luego busca el menor valor contenido en el arreglo y muestra ese menor.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

¿Qué hace la siguiente función en Python?

```
def comprobar(v):
    n = len(v)
    for i in range(n-1):
        if v[i] > v[i+1]:
            return False
    return True
```

Seleccione una:

- a.  
Retorna *True* si el arreglo *v* tomado como parámetro contiene al valor *n*, o retorna *False* en caso contrario.
- b.  
Retorna *True* si el arreglo *v* tomado como parámetro está ordenado de mayor a menor, o retorna *False* en caso contrario.
- c.  
Retorna *True* si el arreglo *v* tomado como parámetro contiene todos sus elementos iguales, o retorna *False* en caso contrario.
- d.  
Retorna *True* si el arreglo *v* tomado como parámetro está ordenado de menor a mayor, o retorna *False* en caso contrario. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Retorna *True* si el arreglo *v* tomado como parámetro está ordenado de menor a mayor, o retorna *False* en caso contrario.

**Pregunta 10**

Correcta

Puntúa 3 sobre 3

¿Qué hace la siguiente función en Python?

```
def generar(v):
    n = len(v)

    ac = 0
    for i in range(n):
        ac += v[i]
    p = ac / n

    c = 0
    for i in range(n):
        if v[i] >= p:
            c += 1

    mp = c * [0]
    idx = 0
    for i in range(n):
        if v[i] >= p:
            mp[idx] = v[i]
            idx += 1

    return mp
```

Seleccione una:

- a.  
Toma un arreglo v como parámetro. Genera un segundo arreglo mp que contiene sólo los elementos de v que son mayores a 10 y retorna el nuevo arreglo mp.
- b.  
Toma un arreglo v como parámetro. Ordena el arreglo v. Finalmente, genera un segundo arreglo mp que contiene todos los elementos de v, y retorna mp (que entonces, será igual al vector v pero ordenado).
- c.  
Toma un arreglo v como parámetro. Genera un segundo arreglo mp que contiene solo los elementos no negativos de v, y retorna el nuevo arreglo mp.
- d.  
Toma un arreglo v como parámetro. Calcula el promedio de los valores contenidos en v. Finalmente, genera un segundo arreglo mp que contiene sólo los elementos de v que son mayores o iguales al promedio y retorna el nuevo arreglo mp. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Toma un arreglo v como parámetro. Calcula el promedio de los valores contenidos en v. Finalmente, genera un segundo arreglo mp que contiene sólo los elementos de v que son mayores o iguales al promedio y retorna el nuevo arreglo mp.

**Comenzado el** domingo, 29 de julio de 2018, 11:16

**Estado** Finalizado

**Finalizado en** lunes, 30 de julio de 2018, 07:01

**Tiempo empleado** 19 horas 45 minutos

**Puntos** 13/13

**Calificación** 10 de 10 (96%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál es la idea principal en la que se basa el algoritmo de ordenamiento de Selección Directa que se presentó en clase? (Suponga que se desea ordenar el arreglo de menor a mayor).

Seleccione una:

a.

Realizar varias pasadas sobre el arreglo, en cada pasada seleccionar el menor valor de los que aún no han sido ordenados, y colocarlo en su casilla definitiva. ✓

¡Correcto!

b.

Realizar varias pasadas sobre el arreglo, en cada pasada comparar a cada elemento con el que le sigue inmediatamente, intercambiarlos si están desordenados y seguir haciendo pasadas en esta forma hasta que el arreglo quede ordenado.

c.

Realizar varias pasadas sobre el arreglo. En cada pasada seleccionar un elemento cualquiera  $x$ . Pasar a la derecha del arreglo todos los valores mayores a  $x$ . Pasar a la izquierda del arreglo todos los valores menores a  $x$ . Finalmente, aplicar la misma idea a cada una de las dos "mitades" así obtenidas y seguir así hasta que ya no pueda seguir obteniendo nuevas "mitades".

d.

Realizar una sola pasada sobre el arreglo, en esa pasada comparar a cada elemento con el que le sigue e intercambiarlos si están invertidos.

¡Correcto!

La respuesta correcta es:

Realizar varias pasadas sobre el arreglo, en cada pasada seleccionar el menor valor de los que aún no han sido ordenados, y colocarlo en su casilla definitiva.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

La siguiente función implementa el algoritmo de ordenamiento por *Selección Directa*, tal como se analizó en clases para ordenar de menor a mayor un arreglo *v* con *n* componentes:

```
def selection_sort(v):
    # ordenamiento por seleccion directa
    n = len(v)
    for i in range(n-1):
        for j in range(i+1, n):
            if v[i] > v[j]:
                v[i], v[j] = v[j], v[i]
```

¿Qué efecto se produciría en la función anterior si la instrucción condicional *if v[i] > v[j]:* fuese reemplazada por *if v[i] < v[j]:*?

Seleccione una:

- a.  
No causaría ningún efecto particular: el arreglo seguiría siendo ordenado de menor a mayor.
- b.  
Provocaría que el arreglo permanezca siempre sin cambio alguno (no será nunca ordenado de forma alguna, ni se modificará nunca su contenido original).
- c.  
Provocará que el programa se interrumpa con un mensaje de error en la primera comparación.
- d.  
El arreglo sería ordenado, pero ahora de mayor a menor. ✓  
¡Correcto!

¡Correcto!

La respuesta correcta es:

El arreglo sería ordenado, pero ahora de mayor a menor.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuántas comparaciones hace el algoritmo de Búsqueda Secuencial para encontrar un valor  $x$  en un arreglo de  $n$  componentes en el *peor caso posible*? (El peor caso es el que obliga a un algoritmo a hacer la máxima cantidad de trabajo. Obviamente, en el caso de la Búsqueda Secuencial ese peor caso se presenta si el valor buscado está exactamente en la última casilla, o bien, si el valor buscado no está en el arreglo).

Seleccione una:

- a.  
Una sola comparación, siempre.
- b.  
 $\log_2(n)$  comparaciones.
- c.  
 $n$  comparaciones. ✓  
¡Correcto!
- d.  
 $n^2$  comparaciones.

¡Correcto!

La respuesta correcta es:  
 $n$  comparaciones.

**Pregunta 4**

Correcta

Puntúa 2 sobre 2

Oportunamente se presentó en clases el algoritmo de búsqueda secuencial, el cual toma un arreglo, busca un valor  $x$  en el mismo casilla por casilla, y retorna el índice de la casilla que lo contiene (si  $x$  está en el arreglo) o retorna -1 si  $x$  no está en el arreglo. Suponga que propone la siguiente variante para el algoritmo de búsqueda secuencial:

```
def linear_search(v, x):
    r = -1
    for i in range(len(v)):
        if x == v[i]:
            r = i

    return r
```

¿Hay algún inconveniente o problema con esta variante?

Seleccione una:

- a.  
La variante propuesta no funciona correctamente: sólo llega a analizar el contenido de la primera casilla.
- b.  
La variante propuesta funciona solamente si el valor  $x$  está en el arreglo (falla si  $x$  no está).
- c.  
Hay un inconveniente: la variante propuesta encuentra el valor buscado (si existía) pero no corta el ciclo de búsqueda al encontrarlo con lo que siempre se hacen  $n$  comparaciones. ✓  
¡Correcto!
- d.  
La variante propuesta funciona correctamente.

¡Correcto!

La respuesta correcta es:

Hay un inconveniente: la variante propuesta encuentra el valor buscado (si existía) pero no corta el ciclo de búsqueda al encontrarlo con lo que siempre se hacen  $n$  comparaciones.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

Oportunamente se presentó en clases el algoritmo de búsqueda secuencial, el cual toma un arreglo, busca un valor  $x$  en el mismo casilla por casilla, y retorna el índice de la casilla que lo contiene (si  $x$  está en el arreglo) o retorna -1 si  $x$  no está en el arreglo. Suponga que propone la siguiente variante para el algoritmo de búsqueda secuencial:

```
def linear_search(v, x):
    n = len(v)
    for i in range(n):
        if x == v[i]:
            return i
    else:
        return -1
```

¿Funciona correctamente esta variante? Si no funciona, ¿cuál es el problema?

Seleccione una:

a.

La variante propuesta funciona correctamente.

b.

La variante propuesta funciona solamente si el valor  $x$  está en el arreglo (falla si  $x$  no está).

c.

La variante propuesta no funciona correctamente: sólo llega a analizar el contenido de la primera casilla. ✓

¡Correcto!

d.

La variante propuesta provoca un error de intérprete: un if no puede tener una instrucción return en cada una de sus ramas.

¡Correcto!

La respuesta correcta es:

La variante propuesta no funciona correctamente: sólo llega a analizar el contenido de la primera casilla.

**Pregunta 6**

Parcialmente correcta

Puntúa 1 sobre 1

¿En cuáles de los siguientes casos es aplicable el algoritmo de búsqueda binaria en un arreglo? (Seleccione todas las respuestas que considere correctas)

Seleccione una o más de una:



a.

El arreglo en el cual se debe realizar la búsqueda está ordenado y permanecerá sin cambios. 

¡Correcto!



b.

El arreglo en el cual se debe realizar la búsqueda está desordenado y no se nos permite ordenarlo.



c.

El arreglo en el cual se debe realizar la búsqueda está desordenado, se nos permite ordenarlo, luego podrá alterarse el contenido (quedando eventualmente desordenado) y debemos realizar varias búsquedas. 

Incorrecto... Si el arreglo se modifica y se puede desordenar, la búsqueda binaria no funciona... y si tiene que volver a ordenarlo cada vez que quiera buscar, el escenario no es aplicable: el tiempo que demore en ordenar es mucho mayor que el tiempo que ganará buscando en forma veloz.



d.

El arreglo en el cual se debe realizar la búsqueda está desordenado, se nos permite ordenarlo, luego permanecerá sin cambios y debemos realizar muchas búsquedas.

Las respuestas correctas son:

El arreglo en el cual se debe realizar la búsqueda está ordenado y permanecerá sin cambios..

El arreglo en el cual se debe realizar la búsqueda está desordenado, se nos permite ordenarlo, luego permanecerá sin cambios y debemos realizar muchas búsquedas.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

Oportunamente se presentó en clases el algoritmo de búsqueda binaria, el cual toma un arreglo ordenado, busca un valor  $x$  en el mismo, y retorna el índice de la casilla que lo contiene (si  $x$  está en el arreglo) o retorna -1 si  $x$  no está en el arreglo. Suponga que propone la siguiente variante para el algoritmo de búsqueda binaria:

```
def binary_search(v, x):
    # busqueda binaria... asume arreglo ordenado...
    izq, der = 0, len(v) - 1
    while izq <= der:
        c = (izq + der) // 2
        if x == v[c]:
            return c
        else:
            return -1

        if x < v[c]:
            der = c - 1
        else:
            izq = c + 1

    return -1
```

¿Funciona correctamente esta variante? Si no funciona, ¿cuál es el problema?

Seleccione una:

a.

La variante propuesta funciona correctamente.

b.

La variante propuesta no funciona correctamente: sólo llega a analizar el contenido de la casilla del centro del arreglo. ✓

¡Correcto!

c.

La variante propuesta provoca un error de intérprete y no llega a arrancar: el segundo if incluido dentro del ciclo nunca puede ejecutarse, ya que las dos ramas del if anterior terminan con una instrucción return.

d.

La variante propuesta funciona solamente si el valor  $x$  está en el arreglo (falla si  $x$  no está).

¡Correcto!

La respuesta correcta es:

La variante propuesta no funciona correctamente: sólo llega a analizar el contenido de la casilla del centro del arreglo.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Oportunamente se presentó en clases el algoritmo de búsqueda binaria, el cual toma un arreglo ordenado, busca un valor  $x$  en el mismo, y retorna el índice de la casilla que lo contiene (si  $x$  está en el arreglo) o retorna -1 si  $x$  no está en el arreglo. Mostramos el algoritmo para dar mejor contexto a la pregunta:

```
def binary_search(v, x):
    # busqueda binaria... asume arreglo ordenado...
    izq, der = 0, len(v) - 1
    while izq <= der:
        c = (izq + der) // 2
        if x == v[c]:
            return c
        if x < v[c]:
            der = c - 1
        else:
            izq = c + 1

    return -1
```

¿Qué pasaría con la función anterior si el arreglo  $v$  contuviese *cadenas de caracteres* en cada casillero (en lugar de números), y el parámetro  $x$  contuviese también una cadena de caracteres?

Seleccione una:

 a.

La función mostrada funcionaría correctamente de todos modos. ✓

¡Correcto!

 b.

La función mostrada provocaría un error en tiempo de ejecución y se interrumpiría el programa al intentar determinar si  $x$  menor o mayor que  $v[i]$ .

 c.

La función mostrada no provocaría que el programa se interrumpa, pero funcionaría en forma incorrecta y retornaría siempre -1 (nunca encontraría la cadena  $x$  buscada).

 d.

La función mostrada funciona solamente si el valor  $x$  está en el arreglo (falla si  $x$  no está, interrumpiendo el programa con un mensaje de error).

¡Correcto!

La respuesta correcta es:

La función mostrada funcionaría correctamente de todos modos.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Qué pasaría con el algoritmo de *Búsqueda Binaria* si el valor buscado  $x$  estuviese repetido más de una vez en el arreglo?

Seleccione una:

- a.  
El algoritmo no provocaría que el programa se interrumpa, pero funcionaría en forma incorrecta y retornaría siempre -1 (nunca encontraría *el valor x buscado*).
- b.  
El algoritmo funcionaría correctamente de todos modos: retornaría el índice de la primera casilla encontrada que contenga a  $x$ .
- c. El algoritmo produciría un error en tiempo de ejecución y se interrumpiría el programa.
- d.  
El algoritmo funciona solamente si el valor  $x$  está en el arreglo (falla si  $x$  no está, interrumpiendo el programa con un mensaje de error).

¡Correcto!

La respuesta correcta es:

El algoritmo funcionaría correctamente de todos modos: retornaría el índice de la primera casilla encontrada que contenga a  $x$ .

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

¿En cuáles de los siguientes casos es aplicable el algoritmo de *Fusión de Arreglos* que se describió en las fichas de clases para producir un tercer arreglo ordenado? (Seleccione todas las respuestas que considere correctas)

Seleccione una o más de una:

 a.

Uno de los arreglos originales debe estar ordenado y el otro arreglo puede estar desordenado.

 b.

El proceso es aplicable sin importar si los dos arreglos originales están ordenados o no, ya que el algoritmo de fusión analizado primero ordena esos dos arreglos, y luego procede a la fusión.

 c.

Los arreglos originales deben estar desordenados.

 d.

Los arreglos originales deben estar ordenados de menor a mayor si se quiere producir un tercer arreglo ordenado de menor a mayor. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Los arreglos originales deben estar ordenados de menor a mayor si se quiere producir un tercer arreglo ordenado de menor a mayor.

---

**Comenzado el** viernes, 17 de agosto de 2018, 13:09

**Estado** Finalizado

**Finalizado en** jueves, 23 de agosto de 2018, 21:28

**Tiempo empleado** 6 días 8 horas

**Puntos** 14/14

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

El siguiente programa crea tres arreglos paralelos para almacenar los legajos, los nombres y los promedios de  $n$  estudiantes y luego los carga por teclado. ¿Qué muestra finalmente este programa al ejecutarlo?

```
__author__ = 'Cátedra de AED'

def read(leg, nom, pro):
    n = len(leg)
    for i in range(n):
        leg[i] = int(input('Legajo: '))
        nom[i] = input('Nombre: ')
        pro[i] = float(input('Promedio: '))

def procesar(leg, nom, pro):
    im = 0
    for i in range(1, len(pro)):
        if pro[i] > pro[im]:
            im = i
    return leg[im], nom[im], pro[im]

def test():
    n = int(input('Cantidad de alumnos: '))
    leg = n * [0]
    nom = n * ['']
    pro = n * [0.0]
    read(leg, nom, pro)

    r = procesar(leg, nom, pro)

    print('Datos del estudiante pedido...')
    print('Legajo:', r[0])
    print('Nombre:', r[1])
    print('Promedio:', r[2])

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

Seleccione una:

a.

Busca el estudiante con menor promedio del arreglo (mediante la función `procesar()`) y muestra sus datos completos.

b.

Busca el estudiante con mayor promedio del arreglo (mediante la función `procesar()`) y muestra el índice de la casilla del arreglo que lo contiene.

- c.  
Busca el estudiante con mayor promedio del arreglo (mediante la función *procesar()*) y muestra sólo el valor de ese promedio.
  
- d.  
Busca el estudiante con mayor promedio del arreglo (mediante la función *procesar()*) y muestra sus datos completos. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Busca el estudiante con mayor promedio del arreglo (mediante la función *procesar()*) y muestra sus datos completos.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

El siguiente programa crea y carga un arreglo *numeros* con *n* números enteros y luego procesa ese arreglo con la función *control()*. ¿Qué hace exactamente esa función al ejecutarse?

```
__author__ = 'Cátedra de AED'

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

def control(numeros):
    n = len(numeros)
    for i in range(n-1):
        if numeros[i] != numeros[i+1]:
            return False
    return True

def test():
    n = int(input('Cantidad de números a cargar: '))
    numeros = n * [0]
    read(numeros)

    print()
    if control(numeros):
        print('El contenido del arreglo es correcto')
    else:
        print('El contenido del arreglo no es correcto')

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

Seleccione una:

- a.  
Retorna True si los números del arreglo no son todos iguales (hay dos o más números diferentes).
- b.  
Retorna True si el arreglo está ordenado de menor a mayor.
- c.  
Retorna True si el arreglo está ordenado de mayor a menor.
- d.  
Retorna True si todos los números del arreglo son iguales. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Retorna True si todos los números del arreglo son iguales.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

El siguiente programa crea y carga un arreglo con n números enteros y luego procesa ese arreglo mediante las funciones `conteo()` y `suma()` ¿Qué hacen exactamente esas dos funciones?

```
__author__ = 'Cátedra de AED'

def read(numeros):
    n = len(numeros)
    for i in range(n):
        numeros[i] = int(input('Valor para la casilla ' + str(i)
+ ': '))

def conteo(numeros):
    c = 0
    for x in numeros:
        if x > 0:
            c += 1
    return c

def suma(numeros):
    s = 0
    n = len(numeros)
    for i in range(0, n, 2):
        s += numeros[i]
    return s

def test():
    n = int(input('Cantidad de números a cargar: '))
    numeros = n * [0]
    read(numeros)

    cmc = conteo(numeros)
    scp = suma(numeros)

    print('Conteo pedido:', cmc)
    print('Suma pedida:', scp)

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

Seleccione una:

- a.

La función `conteo()` cuenta los números mayores a 0 que hay en el arreglo `numeros`, y la función `suma()` acumula los valores del arreglo `numeros`.

- b.

La función `conteo()` cuenta los números mayores a 0 que hay en el arreglo `numeros`, y la función `suma()` acumula los valores contenidos en las casillas con índice par del arreglo `numeros`. ✓

¡Correcto!

c.

La función `conteo()` cuenta los números mayores a 0 que hay en el arreglo `numeros`, y la función `suma()` acumula los valores contenidos en las casillas con índice impar del arreglo `numeros`.

d.

La función `conteo()` cuenta los números mayores a 0 almacenados en casillas con índice impar que hay en el arreglo `numeros`, y la función `suma()` acumula los valores contenidos en las casillas con índice par del arreglo `numeros`.

¡Correcto!

La respuesta correcta es:

La función `conteo()` cuenta los números mayores a 0 que hay en el arreglo `numeros`, y la función `suma()` acumula los valores contenidos en las casillas con índice par del arreglo `numeros`.

**Pregunta 4**

Correcta

Puntúa 2 sobre 2

El siguiente programa crea y carga un arreglo *ventas* con *n* montos de ventas realizadas por un comercio en distintos momentos. ¿Qué hace exactamente el programa completo al ejecutarse?

```
__author__ = 'Cátedra de AED'

def read(ventas):
    n = len(ventas)
    for i in range(n):
        ventas[i] = int(input('Monto de venta[' + str(i) + ']: '))
    )

def sort(ventas):
    n = len(ventas)
    for i in range(n-1):
        for j in range(i+1, n):
            if ventas[i] < ventas[j]:
                ventas[i], ventas[j] = ventas[j], ventas[i]

def display(ventas, cant):
    n = len(ventas)
    if cant > n:
        print('La cantidad de ventas registradas no alcanza para el listado pedido...')
    return

    print('Montos de las', cant, 'ventas pedidas:')
    for i in range(cant):
        print('Monto[' + str(i) + ']:', ventas[i])

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

    sort(ventas)

    print()
    display(ventas, 3)

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

Seleccione una:

a.

Muestra los *cant* mayores montos del arreglo *ventas*, ordenados de menor a mayor.

b.

Muestra siempre todos los montos del arreglo *ventas*, en el mismo orden que tenían en el arreglo *ventas*.

c.

Muestra los *cant* mayores montos del arreglo *ventas*, ordenados de mayor a menor. ✓

¡Correcto!

d.

Muestra siempre todos los montos del arreglo *ventas*, ordenados de mayor a menor.

¡Correcto!

La respuesta correcta es:

Muestra los *cant* mayores montos del arreglo *ventas*, ordenados de mayor a menor.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

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 menor temperatura del arreglo `temp`.

b.

Calcula y retorna la diferencia entre la mayor y la menor temperatura del arreglo `temp`. ✓

¡Correcto!

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 6**

Correcta

Puntúa 1 sobre 1

El siguiente programa crea tres arreglos para cargar y almacenar en ellos los legajos, los nombres y los promedios de n estudiantes. ¿Qué hace finalmente este programa al ejecutarlo?

```
__author__ = 'Cátedra de AED'

def read(leg, nom, pro):
    n = len(leg)
    for i in range(n):
        leg[i] = int(input('Legajo: '))
        nom[i] = input('Nombre: ')
        pro[i] = float(input('Promedio: '))

def search(nom, x):
    n = len(nom)
    for i in range(n):
        if x == nom[i]:
            return i
    return -1

def test():
    n = int(input('Cantidad de alumnos: '))
    leg = n * [0]
    nom = n * ['']
    pro = n * [0.0]
    read(leg, nom, pro)

    x = input('Ingrese el nombre del estudiante a buscar: ')
    ind = search(nom, x)

    if ind != -1:
        print('El estudiante pedido está registrado en la posición', ind, 'y sus datos son:')
        print('Legajo:', leg[ind])
        print('Nombre:', nom[ind])
        print('Promedio:', pro[ind])
    else:
        print('No hay un estudiante con ese nombre...')

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

Seleccione una:

a.

Busca en el arreglo de nombres un estudiante con nombre igual a x, mediante la función `search()` aplicando búsqueda binaria. Si tal nombre existe muestra todos los datos del estudiante, y si no existe se muestra un mensaje avisando de ese hecho.

b.

Busca en el arreglo de nombres un estudiante con nombre igual a  $x$ , mediante la función `search()` aplicando búsqueda secuencial. Si tal nombre existe muestra el índice de la casilla que lo contenía, y si no existe se muestra un mensaje avisando de ese hecho.

c.

Busca en el arreglo de nombres un estudiante con nombre igual a  $x$ , mediante la función `search()` aplicando búsqueda secuencial. Si tal nombre existe muestra todos sus datos del estudiante, y si no existe se muestra un mensaje avisando de ese hecho. ✓

¡Correcto!

d.

Busca en el arreglo de nombres un estudiante con nombre igual a  $x$ , mediante la función `search()` aplicando búsqueda secuencial. Si tal nombre existe muestra el promedio (y sólo el promedio) de estudiante, y si no existe se muestra el valor `None`.

¡Correcto!

La respuesta correcta es:

Busca en el arreglo de nombres un estudiante con nombre igual a  $x$ , mediante la función `search()` aplicando búsqueda secuencial. Si tal nombre existe muestra todos sus datos del estudiante, y si no existe se muestra un mensaje avisando de ese hecho.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

El programa que sigue crea y carga tres arreglos con los legajos, nombres y promedios de n estudiantes. ¿Qué hace concretamente la función **check()** del programa?

```
__author__ = 'Cátedra de AED'

def read(leg, nom, pro):
    n = len(leg)
    for i in range(n):
        leg[i] = int(input('Legajo: '))
        nom[i] = input('Nombre: ')
        pro[i] = float(input('Promedio: '))

def display_all(leg, nom, pro):
    n = len(leg)
    for i in range(n):
        print('Legajo:', leg[i], end=' ')
        print('Nombre:', nom[i], end=' ')
        print('Promedio:', pro[i])

def check(pro):
    for p in pro:
        if p < 4:
            return True
    return False

def test():
    n = int(input('Cantidad de alumnos: '))
    leg = n * [0]
    nom = n * ['']
    pro = n * [0.0]
    read(leg, nom, pro)

    print()
    display_all(leg, nom, pro)

    if check(pro):
        print('Se ha registrado al menos un estudiante aplazado...')
    else:
        print('No se han registrado estudiantes aplazados...')

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

Seleccione una:

- a.

Retorna la suma acumulada de todos los promedios del arreglo *pro*.

- b.

Retorna *True* si el arreglo *pro* contiene alguna casilla valiendo *None*, y retorna *False* si todos los casilleros son diferentes de *None*.

- c.

Retorna *True* si el arreglo *pro* contiene al menos un valor menor a 4, y retorna *False* si todos los promedios del arreglo son mayores o iguales a 4. ✓

¡Correcto!

- d.

Retorna *True* si todos los valores del vector *pro* son mayores o iguales a 4, y *False* si al menos un promedio es menor a 4.

¡Correcto!

La respuesta correcta es:

Retorna *True* si el arreglo *pro* contiene al menos un valor menor a 4, y retorna *False* si todos los promedios del arreglo son mayores o iguales a 4.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Qué hace la siguiente función en Python?

```
def contar(n):
    v = n * [0]

    num = int(input('Ingrese un valor entre 0 y ' + str(n)
+ '(con -1 corta):'))
    while num != -1:
        if 0 <= num < n:
            v[num] += 1
        else:
            print('Error. El número debe ser >= 0 y <',
n)
            num = int(input('Ingrese otro valor entre 0 y ' +
str(n) + '(con -1 corta):'))

    return v
```

Seleccione una:

 a.

Carga por teclado una secuencia de números, y usa el vector v para validar que esos números estén dentro del rango [0, n-1].

 b.

Carga por teclado una secuencia de números, y usa el vector v para almacenar esos números.

 c.

Carga por teclado una secuencia de números, y usa el vector v para contar cuántas veces apareció cada número. ✓

¡Correcto!

 d.

Carga por teclado una secuencia de números, y usa el vector v para acumular esos números.

¡Correcto!

La respuesta correcta es:

Carga por teclado una secuencia de números, y usa el vector v para contar cuántas veces apareció cada número.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

Suponga que la variable *codigos* tomada como parámetro en la siguiente función se usa para almacenar números entre 0 y 24 que indican en qué idioma están escritos los n libros de un conjunto de libros que se tiene que procesar (por ejemplo, 0: español, 1: inglés, 2: italiano, etc.) Asuma que ese arreglo de códigos fue creado y cargado correctamente antes de ser invocada la función. ¿Qué hace entonces la siguiente función?

```
def procesar(codigos):
    n = len(codigos)
    c = 25 * [0]
    for i in range(n):
        id = codigos[i]
        if 0 <= id <= 24:
            c[id] += 1
    return c
```

Seleccione una:

a.

Usa un vector *c* de números enteros, para contar cuántos libros hay en cada idioma, y luego retorna el vector de conteos *c*. ✓

¡Correcto!

b.

Usa un vector *c* de números enteros, para contar cuántos libros hay en cada idioma, y luego retorna el vector original *codigos*.

c.

Usa un vector *c* de números enteros, para contar cuántos libros hay en cada idioma, y luego retorna la conversión a cadena de caracteres del vector de conteos *c*.

d.

Usa un vector *c* de números enteros, para contar cuántos libros hay en cada idioma, y luego retorna el valor más alto del vector de conteos.

¡Correcto!

La respuesta correcta es:

Usa un vector *c* de números enteros, para contar cuántos libros hay en cada idioma, y luego retorna el vector de conteos *c*.

**Pregunta 10**

Correcta

Puntúa 2 sobre 2

El siguiente programa crea y carga dos arreglos paralelos *destinos* y *montos* con los datos de  $n$  llamadas telefónicas registradas para un cliente en distintos momentos. En el arreglo *destinos* se almacenan los códigos de los lugares de destino de cada llamada y se asumen que esos destinos se representan con números entre 0 y 24 (por ejemplo: 0: Estados Unidos, 1: Brasil, etc.) El arreglo *montos* almacena el costo de cada llamada. ¿Qué hace exactamente el programa completo al ejecutarse?

```

__author__ = 'Cátedra de AED'

def read(destinos, montos):
    n = len(destinos)
    for i in range(n):
        destinos[i] = int(input('Código de destino de la llamada
(valor entre 0 y 24 por favor): '))
        montos[i] = float(input('Monto: '))

def process(destinos, montos):
    n = len(destinos)

    s = 25 * [0]
    for i in range(n):
        d = destinos[i]
        if 0 <= d <= 24:
            s[d] += montos[i]

    return s

def display(s):
    print('Listado solicitado de llamadas...')

    m = len(s)
    for i in range(m):
        if s[i] != 0:
            print('Destino:', i, 'Total:', s[i])

def test():
    n = int(input('Cantidad de llamadas: '))
    destinos = n * [0]
    montos = n * [0.0]
    read(destinos, montos)

    s = process(destinos, montos)

    print()
    display(s)

if __name__ == '__main__':
    test()

```

Seleccione una:

a.

Muestra una línea única con el monto total acumulado entre todas las llamadas realizadas.

- b.  
Muestra la cantidad de llamadas realizadas para cada uno de los 25 posibles destinos (sólo para los destinos cuya cantidad de llamadas sea diferente de cero).
  - c.  
Muestra los montos acumulados en llamadas realizadas para cada uno de los 25 posibles destinos (incluyendo en el listado los destinos cuyo monto acumulado sea cero).
  - d.  
Muestra los montos acumulados en llamadas realizadas para cada uno de los 25 posibles destinos (sólo para los destinos cuyo monto acumulado sea diferente de cero). ✓
- ¡Correcto!

¡Correcto!

La respuesta correcta es:

Muestra los montos acumulados en llamadas realizadas para cada uno de los 25 posibles destinos (sólo para los destinos cuyo monto acumulado sea diferente de cero).

---

**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 *códigos* en los que se han almacenado datos de socios de un club deportivo. En el primer arreglo 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 *códigos* 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(códigos):  
    vc = 10 * [0]  
    for d in códigos:  
        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.

 d.

Podría ocurrir que entre un código 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 código 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 arreglo 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 interrumpirá.

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 correctamente 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 verdadera 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 vehículo:', c)
    else:
        print('No está registrado ese vehículo')
```

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. ✓
- 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 ✓
- c. Holamundootravez

- d. Hola mundo otra vez

¡Correcto!

La respuesta correcta es:

Hola\*\*mundo\*\*otra\*\*vez

**Comenzado el** lunes, 10 de septiembre de 2018, 14:18

**Estado** Finalizado

**Finalizado en** lunes, 10 de septiembre de 2018, 14:41

**Tiempo empleado** 22 minutos 45 segundos

**Puntos** 13/15

**Calificación** 9 de 10 (87%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál de la siguientes afirmaciones es *incorrecta* en relación a las características de los arreglos en Python?

Seleccione una:

a.

Un arreglo (variable de tipo *list*) en Python puede contener valores de cualquier tipo, incluso entremezclados.

b.

Siempre se puede usar la función `len()` para determinar la longitud o cantidad de componentes de un arreglo (variable de tip *list*)

c.

En Python sólo se pueden crear arreglos (variables de tipo *list*) de dimensión 1 o de dimensión 2. 

¡Correcto! Esta afirmación es justamente falsa: en Python una variable de tipo *list* puede contener a otras anidadas, permitiendo que el programador cree así arreglos de la dimensión que necesite.

d.

Los índices de un arreglo (variable de tipo *list*) en Pyhton, comienzan siempre desde el número 0 (si no se consideran índices negativos)

¡Correcto!

La respuesta correcta es:

En Python sólo se pueden crear arreglos (variables de tipo *list*) de dimensión 1 o de dimensión 2.

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

Asuma que se ha creado una matriz *reg* en la que cada fila representa un año calendario desde el 2000 en adelante (la fila 0 representa al año 2000, la 1 al 2001 y así sucesivamente) y cada columna representa una de las carreras disponibles (numeradas desde 0 en adelante). Suponga que en casillero *reg[i][j]* se ha almacenado un número que indica cuántos alumnos se inscribieron en la materia *j* en el año *i*, y suponga finalmente que esa matriz fue correctamente creada y cargada . ¿Qué hace la siguiente función, suponiendo que toma como parámetro a la citada matriz *reg*?

```
def procesar(reg):  
    anios = len(reg)  
    carreras = len(reg[0])  
    fm, cm = 0, 0  
    for i in range(anios):  
        for j in range(carreras):  
            if reg[i][j] > rge[fm][cm]:  
                fm = i  
                cm = j  
    return fm
```

Seleccione una:

- a.  
Busca en toda la matriz el casillero con el mayor número de inscriptos y retorna el índice de columna que corresponde a ese casillero(o sea, retorna el número de la carrera para la que ocurrió ese caso).
- b.  
Busca en toda la matriz el casillero con el menor número de inscriptos y retorna el índice de fila que corresponde a ese casillero (o sea, retorna el año en que ocurrió ese caso).
- c.  
Busca en toda la matriz el casillero que tenga el menor número de inscriptos y retorna el índice de columna que corresponde a ese casillero (o sea, retorna el número de la carrera para la que ocurrió ese caso).
- d.  
Busca en toda la matriz el casillero con el mayor número de inscriptos y retorna el índice de fila que corresponde a ese casillero (o sea, retorna el año en que ocurrió ese caso). ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Busca en toda la matriz el casillero con el mayor número de inscriptos y retorna el índice de fila que corresponde a ese casillero (o sea, retorna el año en que ocurrió ese caso).

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Qué hace el siguiente segmento de instrucciones en Python?

```
p = []
for i in range(3):
    p.append([])
    for j in range(2):
        p[i].append([])
        for k in range(4):
            p[i][j].append([])

p[2][1][3] = 'Prueba'
print(p)
```

Seleccione una:

 a.

Crea un arreglo *p* de tres dimensiones, y almacena una cadena en el último casillero de ese arreglo (el casillero de índices más altos de cada dimensión posible). ✓

¡Correcto!

 b.

Crea un arreglo *p* de cuatro dimensiones, y almacena una cadena en el último casillero de ese arreglo (el casillero de índices más altos de cada dimensión posible).

 c.

Crea un arreglo *p* de tres dimensiones, y almacena una cadena en el primer casillero de ese arreglo (el casillero de índices más bajos de cada dimensión posible).

 d.

Lanza un error de intérprete: en Python no se pueden crear arreglos (variables de tipo *list*) de dimensión mayor a 2.

¡Correcto!

La respuesta correcta es:

Crea un arreglo *p* de tres dimensiones, y almacena una cadena en el último casillero de ese arreglo (el casillero de índices más altos de cada dimensión posible).

**Pregunta 4**

Incorrecta

Puntúa 0 sobre 1

Suponga la función `test()` (que se muestra más abajo) recibe como parámetro un arreglo bidimensional `mat` ya creado con componentes de tipo `int`.

¿Qué hace concretamente esa función `test()`?

```
def test(mat):
    fils = len(mat)
    cols = len(mat[0])
    for k in range(cols):
        ac = 0
        for g in range(fils):
            ac += mat[k][g]
        p = ac / fils
        print('Indice:', k, '- Promedio:', p)
```

Seleccione una:

a.

Lo que ocurre con este método depende de la cantidad  $n$  de filas y la cantidad  $m$  de columnas que tiene la matriz. Si la matriz es cuadrada ( $n == m$ ) entonces mostrará por pantalla un listado con el promedio de cada *columna*. Si la matriz no es cuadrada ( $n != m$ ) entonces al ejecutar este método se *provocará una excepción por índice fuera de rango*.

b.

Lo que ocurre con este método depende de la cantidad  $n$  de filas y la cantidad  $m$  de columnas que tiene la matriz. Si la matriz es cuadrada ( $n == m$ ) entonces mostrará por pantalla un listado con el promedio de cada *fila*. Si la matriz no es cuadrada ( $n != m$ ) entonces al ejecutar este método se *provocará una excepción por índice fuera de rango*.

c.

Muestra por pantalla un listado con el promedio de cada fila.

d.

Muestra por pantalla un listado con el promedio de cada columna. X

Incorrecto... puede llegar a hacer eso... pero depende de un análisis un poco más profundo...

Incorrecto... revise las fichas de clase sobre arreglos de una o más dimensiones y sus notas de clase... Revise lo que ocurre cuando intenta acceder a un casillero que no existe...

La respuesta correcta es:

Lo que ocurre con este método depende de la cantidad  $n$  de filas y la cantidad  $m$  de columnas que tiene la matriz. Si la matriz es cuadrada ( $n == m$ ) entonces mostrará por pantalla un listado con el promedio de cada *fila*. Si la matriz no es cuadrada ( $n != m$ ) entonces al ejecutar este método se *provocará una excepción por índice fuera de rango*.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

Asuma que se quiere definir una función `registrar()` como la que se muestra más abajo, en la cual se necesita crear una matriz `reg` de forma que en esa matriz cada fila represente un año calendario desde el 2000 en adelante (la fila 0 representaría al año 2000, la 1 al 2001 y así sucesivamente) y cada columna represente una de las carreras disponibles en una universidad (numeradas desde 0 en adelante). La idea es que una vez creada esa matriz, debe usarse para almacenar en cada casillero una cierta cantidad fija de inscriptos ¿Hay algún problema con la siguiente función en Python, asumiendo que `a` es la cantidad de filas y `c` es la cantidad de columnas?

```
def registrar(a, c):
    reg = []
    for i in range(a):
        for j in range(c):
            reg[i][j] = 50

    return reg
```

Seleccione una:

a.

No, no hay problema alguno. La función crea la matriz pedida, e inicializa cada casillero con el valor 50.

b.

Sí, hay un problema: los índices `i` y `j` están al revés cuando se usaron como índices para entrar a una casilla de la matriz.

c.

Sí, hay un problema: al ejecutar la función de producirá un error de intérprete y el programa se interrumpirá, debido a que la matriz en realidad nunca fue creada.



¡Correcto!

d.

Sí, hay un problema pero es menor: la función crea efectivamente la matriz pedida, pero luego inicializa cada casillero con el mismo valor, lo cual no tiene sentido práctico.

¡Correcto!

La respuesta correcta es:

Sí, hay un problema: al ejecutar la función de producirá un error de intérprete y el programa se interrumpirá, debido a que la matriz en realidad nunca fue creada.

**Pregunta 6**

Correcta

Puntúa 2 sobre 2

Suponga función `test()` (que se muestra más abajo) toma como parámetro una matriz `mat` ya definida (y otras dos variables `x` e `y`) y correctamente cargada, con elementos de tipo `int`. ¿Qué hace concretamente esta función?

```
def test(mat, x, y):
    n = len(mat)
    m = len(mat[0])
    if x < 0 or x >= n:
        return False
    if y < 0 or y >= n:
        return False
    if x == y:
        return True

    for k in range(m):
        mat[x][k], mat[y][k] = mat[y][k], mat[x][k]

    return True
```

Seleccione una:

- a.  
Intercambia los valores contenidos en la columna `x` con los valores contenidos en la columna `y`. Retorna `True` si el intercambio pudo hacerse (o si `x` era igual a `y`); y retorna `False` en caso contrario.
- b.  
Comprueba si alguno de los valores contenidos en la fila `x` es igual a alguno de los valores contenidos en la fila `y`. Retorna `True` en ese caso (o si `x` era igual a `y`); y retorna `False` en caso contrario.
- c.  
Intercambia los valores contenidos en la fila `x` con los valores contenidos en la fila `y`. Retorna `True` si el intercambio pudo hacerse (o si `x` era igual a `y`); y retorna `False` en caso contrario. ✓  
¡Correcto!
- d.  
Comprueba si la matriz es cuadrada. En caso de serlo, intercambia los valores contenidos en la fila `x` con los valores contenidos en la columna `y`. Retorna `True` si el intercambio pudo hacerse y retorna `False` en caso contrario.

¡Correcto!

La respuesta correcta es:

Intercambia los valores contenidos en la fila `x` con los valores contenidos en la fila `y`. Retorna `True` si el intercambio pudo hacerse (o si `x` era igual a `y`); y retorna `False` en caso contrario.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

Suponga la función `test()` (que se muestra más abajo) que toma como parámetro una matriz `mat` ya definida y cargada con elementos de tipo `int`. ¿Qué hace concretamente la función `test()`?

```
def test(mat):
    n = len(mat)
    m = len(mat[0])
    r = [n*[0] for k in range(m)]

    for i in range(n):
        for j in range(m):
            r[j][i] = mat[i][j]

    return r
```

Seleccione una:

- a.  
Retorna una copia de la matriz `mat`.
- b.  
Retorna la *máxima matriz cuadrada que puede obtenerse a partir de mat*. Si `mat` ya era cuadrada, retorna una copia de la misma matriz `mat`. Si `mat` era rectangular, crea una nueva matriz con la máxima cantidad de filas y columnas de `mat` que puedan usarse para que esta nueva matriz sea cuadrada, y retorna esa matriz.
- c.  
Obtiene y retorna la *matriz traspuesta de mat* (la *transpuesta* de una matriz `mat` de orden  $n*m$  es igual a otra matriz `r` de orden  $m*n$ , pero de forma tal que las filas de `mat` son ahora las columnas de `r`). ✓  
¡Correcto!
- d.  
Retorna *una nueva matriz, en la cual los valores originales de las filas de mat se invierten*: en cada fila, el valor que en `mat` estaba en la columna 0 aparece en la última columna de la nueva matriz, el valor que estaba en la columna 1 de `mat` aparece en la anteúltima de la nueva matriz, y así sucesivamente.

¡Correcto!

La respuesta correcta es:

Obtiene y retorna la *matriz traspuesta de mat* (la *transpuesta* de una matriz `mat` de orden  $n*m$  es igual a otra matriz `r` de orden  $m*n$ , pero de forma tal que las filas de `mat` son ahora las columnas de `r`).

**Pregunta 8**

Correcta

Puntúa 2 sobre 2

Suponga la función `test()` (que se muestra más abajo) que toma como parámetro una matriz `mat` ya creada y cargada con elementos `int`. ¿Qué hace concretamente la función `test()`?

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

    r = m * [0]
    for i in range(m):
        ac = 0
        for j in range(n):
            ac += mat[j][i]
        r[i] = ac / n

    return r
```

Seleccione una:

a.

Obtiene el valor promedio de cada fila, y retorna un vector con esos promedios.

b.

Obtiene el valor promedio de cada columna, y retorna un vector con esos promedios. ✓

¡Correcto!

c.

Obtiene el valor mayor de cada columna, y retorna un vector con esos mayores.

d.

Obtiene el valor promedio de la matriz, y retorna ese valor.

¡Correcto!

La respuesta correcta es:

Obtiene el valor promedio de cada columna, y retorna un vector con esos promedios.

**Pregunta 9**

Incorrecta

Puntúa 0 sobre 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 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.

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 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. 

Incorrecto... es cierto que comprueba algo en la diagonal, pero no exactamente lo que marcó aquí...

d.

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.

Revise la Fichas 16, sección de Temas Avanzados...

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 10**

Correcta

Puntúa 1 sobre 1

Suponga la función `test()` (que se muestra más abajo) que recibe como parámetro una matriz `mat` de elementos de tipo `int` (además de otro parámetro `x`) ¿Qué hace concretamente la función?

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

    if x < 0 or x >= n:
        print('Imposible mostrar lo pedido...')
        return

    for k in range(m):
        print('Valor:', mat[x][k])
```

Seleccione una:

- a.  
Muestra en consola de salida la fila cuyo índice es `x`. ✓  
¡Correcto!
- b.  
Muestra en consola de salida la columna cuyo índice es `x`.
- c.  
Muestra en consola de salida la matriz completa.
- d.  
Muestra en consola de salida la última fila de la matriz.

¡Correcto!

La respuesta correcta es:

Muestra en consola de salida la fila cuyo índice es `x`.

**Comenzado el** lunes, 10 de septiembre de 2018, 21:01

**Estado** Finalizado

**Finalizado en** lunes, 10 de septiembre de 2018, 21:30

**Tiempo empleado** 29 minutos 53 segundos

**Calificación** 9 de 10 (92%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál es la principal característica de todos los métodos de ordenamiento conocidos como métodos simples o directos?

Seleccione una:

a.

Tienen muy mal rendimiento en tiempo de ejecución si el tamaño n del arreglo es pequeño, y un rendimiento aceptable si n es grande o muy grande.

b.

Tienen muy mal rendimiento en tiempo de ejecución, cualquiera sea el tamaño n del arreglo.

c.

Tienen muy buen rendimiento en tiempo de ejecución, cualquiera sea el tamaño n del arreglo.

d.

Tienen muy mal rendimiento en tiempo de ejecución si el tamaño n del arreglo es grande o muy grande, y un rendimiento aceptable si n es pequeño. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Tienen muy mal rendimiento en tiempo de ejecución si el tamaño n del arreglo es grande o muy grande, y un rendimiento aceptable si n es pequeño.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes describe mejor la idea de funcionamiento en la que está basado el algoritmo conocido como ordenamiento por *Inserción Simple* para ordenar un arreglo de n componentes?

Seleccione una:

- a.  
Reacomodar los n elementos del arreglo en forma aleatoria, controlar si quedó ordenado, y en caso de negativo, volver a reacomodarlos en forma aleatoria, continuando así hasta que en algún momento se obtenga un arreglo ordenado...
- b.  
Realizar n pasadas, de forma que en cada una se determine el menor de los elementos analizados, y llevar ese menor a la casilla pivot.
- c.  
Suponer que el arreglo tiene un subconjunto inicialmente ordenado que contiene sólo al primer elemento, luego realizar n pasadas, de forma que en cada una agregue el siguiente elemento al grupo que está ordenado. ✓  
¡Correcto!
- d.  
Realizar n pasadas, de forma que en cada una se compare a cada elemento con el siguiente, logrando que en cada pasada los mayores vaya acomodándose al final del arreglo.

¡Correcto!

La respuesta correcta es:

Suponer que el arreglo tiene un subconjunto inicialmente ordenado que contiene sólo al primer elemento, luego realizar n pasadas, de forma que en cada una agregue el siguiente elemento al grupo que está ordenado.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes describe mejor la idea de funcionamiento en la que está basado el algoritmo conocido como ordenamiento por *Intercambio Directo* u *Ordenamiento de Burbuja* para ordenar de menor a mayor un arreglo de n componentes?

Seleccione una:

a.

Realizar n pasadas, de forma que en cada una se compare a cada elemento con el siguiente, logrando que en cada pasada los mayores vayan acomodándose al final del arreglo. ✓

¡Correcto!

b.

Suponer que el arreglo tiene un subconjunto inicialmente ordenado que contiene sólo al primer elemento, luego realizar n pasadas, de forma que en cada una agregue el siguiente elemento al grupo que está ordenado.

c.

Reacomodar los n elementos del arreglo en forma aleatoria, controlar si quedó ordenado, y en caso negativo, volver a reacomodarlos en forma aleatoria, continuando así hasta que en algún momento se obtenga un arreglo ordenado...

d.

Realizar n pasadas, de forma que en cada una se determine el menor de los elementos analizados, y llevar ese menor a la casilla pivot.

¡Correcto!

La respuesta correcta es:

Realizar n pasadas, de forma que en cada una se compare a cada elemento con el siguiente, logrando que en cada pasada los mayores vayan acomodándose al final del arreglo.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes describe mejor la idea de funcionamiento en la que está basado el algoritmo conocido como ordenamiento por *Selección Simple* o *Selección Directa* para ordenar un arreglo de n componentes?

Seleccione una:

- a.  
Reacomodar los n elementos del arreglo en forma aleatoria, controlar si quedó ordenado, y en caso de negativo, volver a reacomodarlos en forma aleatoria, continuando así hasta que en algún momento se obtenga un arreglo ordenado...
- b.  
Realizar n pasadas, de forma que en cada una se compare a cada elemento con el siguiente, logrando que en cada pasada los mayores vaya acomodándose al final del arreglo.
- c.  
Realizar n pasadas, de forma que en cada una se determine el menor de los elementos analizados, y llevar ese menor a la casilla pivot. ✓  
¡Correcto!
- d.  
Suponer que el arreglo tiene un subconjunto inicialmente ordenado que contiene sólo al primer elemento, luego realizar n pasadas, de forma que en cada una agregue el siguiente elemento al grupo que está ordenado.

¡Correcto!

La respuesta correcta es:

Realizar n pasadas, de forma que en cada una se determine el menor de los elementos analizados, y llevar ese menor a la casilla pivot.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Por qué motivo el algoritmo Bubblesort para ordenamiento de un arreglo usa una bandera de corte en la versión presentada en las fichas de clase?

Seleccione una:

a.

La bandera de corte se usa para determinar si el ordenamiento debe hacerse de menor a mayor (bandera = True) o de mayor a menor (bandera = False)

b.

La bandera de corte se usa para garantizar que el arreglo quede ordenado.

c.

La bandera de corte se usa para terminar el proceso apenas se detecte que en la pasada actual no hubo intercambios, para ahorrar tiempo. ✓

¡Correcto!

d.

No es cierto que la versión vista en clases use una bandera de corte.

¡Correcto!

La respuesta correcta es:

La bandera de corte se usa para terminar el proceso apenas se detecte que en la pasada actual no hubo intercambios, para ahorrar tiempo.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes es el creador del famoso algoritmo de ordenamiento conocido como Quicksort?

Seleccione una:

- a.  
J. W. J. Williams
- b.  
Donald Shell
- c.  
Edsger Wybe Dijkstra
- d.  
Charles Antony Richard Hoare ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Charles Antony Richard Hoare

**Pregunta 7**

Parcialmente  
correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características **correctas** del algoritmo *Shell Sort*?  
(Más de una puede ser cierta... marque TODAS las que considere válidas)

Seleccione una o más de una:

a.

El algoritmo Shell Sort consiste en una mejora del algoritmo de Selección Directa, basada en buscar iterativamente el menor (o el mayor) entre los elementos que quedan en el vector, para llevarlo a su posición correcta, pero de forma que la búsqueda del menor en cada vuelta se haga en tiempo logarítmico.

b.

Una muy buena serie de incrementos decrecientes a usar, es la serie  $h = \{\dots 16, 8, 4, 2, 1\}$

c.

El algoritmo Shell Sort es complejo de analizar para determinar su rendimientos en forma matemática, ya que ese rendimiento depende fuertemente de la serie de incrementos decreciente que se haya seleccionado.

d.

El algoritmo Shell Sort consiste en una mejora del algoritmo de Inserción Directa (o Inserción Simple), consistente en armar suconjuntos ordenados con elementos a distancia  $h > 1$  en las primeras fases, y terminar con  $h = 1$  en la última. ✓

¡Correcto!

Las respuestas correctas son:

El algoritmo Shell Sort consiste en una mejora del algoritmo de Inserción Directa (o Inserción Simple), consistente en armar suconjuntos ordenados con elementos a distancia  $h > 1$  en las primeras fases, y terminar con  $h = 1$  en la última.,

El algoritmo Shell Sort es complejo de analizar para determinar su rendimientos en forma matemática, ya que ese rendimiento depende fuertemente de la serie de incrementos decreciente que se haya seleccionado.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Cuál es el problema si en el algoritmo *Shellsort* se elige una serie de incrementos decrecientes de la forma { ..., 16, 8, 4, 2, 1} ?

Seleccione una:

a.

El arreglo no quedará ordenado al final.

b.

Ningún problema: esa serie es tan buena como cualquier otra.

c.

Los subconjuntos analizados contendrán casi los mismos elementos cuando la distancia usada sea cada vez menor, sin garantías de lograr una buena organización del arreglo antes de la última pasada. ✓

¡Correcto!

d.

No sólo no hay ningún problema, sino que esa serie es la mejor posible para el algoritmo *Shellsort*.

¡Correcto!

La respuesta correcta es:

Los subconjuntos analizados contendrán casi los mismos elementos cuando la distancia usada sea cada vez menor, sin garantías de lograr una buena organización del arreglo antes de la última pasada.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

Sabemos que en el *algoritmo de Shell* se termina haciendo una última pasada sobre el arreglo con incremento de comparación  $h = 1$  ¿Cuál de las siguientes es cierta respecto de esa última pasada con  $h = 1$ ?

Seleccione una:

- a.

La pasada con  $h = 1$  es obligatoria pero no es necesario que sea la última.

- b.

Con  $h = 1$  el algoritmo se convierte en un ordenamiento por inserción simple, y sólo entonces garantiza que el arreglo quede ordenado. ✓

¡Correcto!

- c.

Con  $h = 1$  el algoritmo sólo controla si el arreglo está ya ordenado, y en caso de no estarlo relanza el proceso con otra sucesión de valores  $h$ .

- d.

No es obligatorio que lo haga, pero favorece un ordenamiento más rápido.

**¡Correcto!**

La respuesta correcta es:

Con  $h = 1$  el algoritmo se convierte en un ordenamiento por inserción simple, y sólo entonces garantiza que el arreglo quede ordenado.

**Pregunta 10**

Parcialmente  
correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características **correctas** del algoritmo *Heap Sort*?  
(Más de una puede ser cierta... marque TODAS las que considere válidas)

Seleccione una o más de una:

a.

El algoritmo Heap Sort se basa en encontrar sucesivamente el menor (o el mayor) de entre los elementos que quedan, para llevar ese valor a su casillero final, pero de forma que la búsqueda del menor (o el mayor) en cada vuelta se haga en forma muy veloz.

b.

El algoritmo Heap Sort arma el heap o grupo de ordenamiento con el que se ordena el vector, pero lo hace en el mismo vector, sin usar memoria extra. ✓

¡Correcto!

c.

El algoritmo Heap Sort utiliza una cantidad de memoria adicional igual al tamaño del arreglo, para armar el heap o grupo de ordenamiento con el que se ordena el vector.

d.

El algoritmo Heap Sort tiene es muy eficiente en tiempo de ejecución, tanto para el caso promedio como para el peor caso. ✓

¡Correcto! Por ahora, no se preocupe del significado profundo de "caso promedio" o "peor caso"...

Revise la Ficha 17, sección de Temas Avanzados...

Ha seleccionado correctamente 2.

Las respuestas correctas son:

El algoritmo Heap Sort tiene es muy eficiente en tiempo de ejecución, tanto para el caso promedio como para el peor caso.,

El algoritmo Heap Sort arma el heap o grupo de ordenamiento con el que se ordena el vector, pero lo hace en el mismo vector, sin usar memoria extra.,

El algoritmo Heap Sort se basa en encontrar sucesivamente el menor (o el mayor) de entre los elementos que quedan, para llevar ese valor a su casillero final, pero de forma que la búsqueda del menor (o el mayor) en cada vuelta se haga en forma muy veloz.

---

**Comenzado el** lunes, 17 de septiembre de 2018, 22:43

**Estado** Finalizado

**Finalizado en** lunes, 17 de septiembre de 2018, 23:03

**Tiempo empleado** 20 minutos 25 segundos

**Puntos** 25/25

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere almacenar en un programa los datos de un *estudiante*, y para representar a ese estudiante se usa una *tupla* en la que el primer casillero guardará el legajo, el segundo el nombre y el tercero el promedio del estudiante. Suponga que parte del programa incluye una secuencia de instrucciones como la siguiente, en la cual se crea la tupla, se muestra su contenido, se cambia el nombre del estudiante y luego se muestran los datos modificados:

```
est = 12345, 'Juan', 8.76
print('Datos del estudiante:', est)

est[1] = 'Pedro'
print('Datos modificados del estudiante:', est)
```

¿Es correcto este enfoque o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.

No es correcto. Los elementos de una tupla no pueden accederse usando índices.

b.

Sí. Es correcto y el esquema mostrado funciona sin problemas.

c.

No es correcto. No se pueden almacenar valores de tipos diferentes en una tupla.

d.

No es correcto. Una tupla podría usarse para representar al estudiante, pero no permitirá modificar ningún componente al ser una secuencia de tipo inmutable. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

No es correcto. Una tupla podría usarse para representar al estudiante, pero no permitirá modificar ningún componente al ser una secuencia de tipo inmutable.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características básicas y generales de un *registro*? (Más de una respuesta puede ser válida. Marque todas las que considere correctas).

Seleccione una o más de una:

- a.

Un registro es un conjunto mutable de datos que pueden ser de tipos diferentes. ✓

¡Correcto!

- b.

Los elementos individuales de un registro se acceden por medio de un *nombre* o *identificador* declarado por el programador, en lugar de hacerlo por medio de índices. ✓

¡Correcto!

- c.

Un registro es un conjunto inmutable de datos que pueden ser de tipos diferentes.

- d.

Los elementos individuales de un registro se designan con el nombre genérico de *campos* o *atributos*. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Un registro es un conjunto mutable de datos que pueden ser de tipos diferentes.,

Los elementos individuales de un registro se designan con el nombre genérico de *campos* o *atributos*.,

Los elementos individuales de un registro se acceden por medio de un *nombre* o *identificador* declarado por el programador, en lugar de hacerlo por medio de índices.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. ¿Hay algún problema con el programa siguiente?

```
class Libro:  
    pass  
  
  
    def init(cod, nom, aut):  
        libro = Libro()  
        libro.codigo = cod  
        libro.titulo = nom  
        libro.autor = aut  
  
        return libro  
  
  
    def write(libro):  
        print('Datos del libro:')  
        print('Código:', libro.codigo, ' - Título:', libro.titulo, ' - Autor:', libro.autor)  
  
  
    def test():  
        lib1 = init(2345, 'El Aleph', 'Jorge Luis Borges')  
        lib2 = init(1267, 'Rayuela', 'Julio Cortázar')  
        lib3 = init(1928, 'El Túnel', 'Ernesto Sábato')  
  
        write(lib1)  
        write(lib2)  
        write(lib3)  
  
  
    if __name__ == '__main__':  
        test()
```

Seleccione una:

a.

No es correcto. La función `init()` está creando nuevamente un registro vacío en su bloque de acciones (al hacer `libro = Libro()`). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en `test()`. Cuando se invoque a la función `write()` se producirá entonces un error de intérprete porque intentará mostrar campos que no existen.

b.

No es correcto. Falta la creación del registro vacío en la función `test()` antes de invocar a `init()`.

c.

Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

d.

No es correcto. La función `init()` no puede invocar a la función constructora (sólo puede invocarse en la función `test()` o en cualquier función que sea usada como función de arranque del programa).

¡Correcto!

La respuesta correcta es:

Sí. Es correcto y el esquema mostrado funciona sin problemas.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. ¿Hay algún problema con el programa siguiente?

```
class Libro:  
    pass  
  
    def init(libro, cod, nom, aut):  
        libro = Libro()  
        libro.codigo = cod  
        libro.titulo = nom  
        libro.autor = aut  
  
    def write(libro):  
        print('Datos del libro:')  
        print('Código:', libro.codigo, ' - Título:', libro.titulo, ' - Autor:', libro.autor)  
  
    def test():  
        lib1 = Libro()  
        init(lib1, 2345, 'El Aleph', 'Jorge Luis Borges')  
  
        lib2 = Libro()  
        init(lib2, 1267, 'Rayuela', 'Julio Cortázar')  
  
        lib3 = Libro()  
        init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')  
  
        write(lib1)  
        write(lib2)  
        write(lib3)  
  
    if __name__ == '__main__':  
        test()
```

Seleccione una:

a.

No es correcto. La función *init()* no puede invocar a la función constructora (sólo puede invocarse en la función *test()* o en cualquier función que sea usada como función de arranque del programa).

b.

Sí. Es correcto y el esquema mostrado funciona sin problemas.

c.

No es correcto. La función *init()* está creando nuevamente un registro vacío en su bloque de acciones (al hacer *libro* = *Libro()*). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en *test()*. Cuando se invoque a la función *write()* se producirá entonces un error de intérprete porque intentará mostrar campos que no existen. ✓

¡Correcto!

d.

No es correcto. Si la función *init()* está creando el registro (al hacer *libro* = *Libro()*) entonces no debe crearse el registro también en *test()*. En *test()* es suficiente con invocar a *init()* para crear cada registro.

¡Correcto!

La respuesta correcta es:

No es correcto. La función *init()* está creando nuevamente un registro vacío en su bloque de acciones (al hacer *libro* = *Libro()*). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en *test()*. Cuando se invoque a la función *write()* se producirá entonces un error de intérprete porque intentará mostrar campos que no existen.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Suponga el siguiente script elemental en el que se crea un tipo registro llamado *Libro*, y luego una variable de ese tipo:

```
class Libro:  
    pass  
  
lib = Libro()
```

¿Qué es lo que hace la función *Libro()* en este script?

Seleccione una:

a.

Define la variable *lib* y la deja asignada con el valor *None*.

b.

El script es incorrecto: la función *Libro()* no existe y se lanzará un error de intérprete.

c.

Crea un registro vacío de tipo *Libro*, lo ubica en memoria y retorna la dirección donde quedó ubicado. Esta dirección se asigna en la variable *lib*, con la cual luego se manejará el registro. ✓

¡Correcto!

d.

Retorna la dirección 0 (o dirección *nula*), que quedará asignada en la variable *lib*. No se crea ningún registro.

¡Correcto!

La respuesta correcta es:

Crea un registro vacío de tipo *Libro*, lo ubica en memoria y retorna la dirección donde quedó ubicado. Esta dirección se asigna en la variable *lib*, con la cual luego se manejará el registro.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. Suponga además que una vez creadas las variables *lib1*, *lib2* y *lib3* se quiere crear una nueva variable *lib4* que contenga los mismos datos que *lib2*, pero de tal forma que ambas se mantengan independientes: al modificar cualquier dato en una de las dos, no debe cambiar nada en la otra. ¿Hay algún problema con el programa siguiente, en base a estos requerimientos?

```
class Libro:  
    pass  
  
def init(libro, cod, nom, aut):  
    libro.codigo = cod  
    libro.titulo = nom  
    libro.autor = aut  
  
def write(libro):  
    print('Datos del libro:')  
    print('Código:', libro.codigo, ' - Título:', libro.titulo, ' - Autor:', libro.autor)  
  
def test():  
  
    lib1 = Libro()  
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borges')  
  
    lib2 = Libro()  
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')  
  
    lib3 = Libro()  
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')  
  
    write(lib1)  
    write(lib2)  
    write(lib3)  
  
    lib4 = Libro()  
    init(lib4, lib2.codigo, lib2.titulo, lib2.autor)  
    lib4.autor = 'Adolfo Bioy Casares'  
  
    write(lib2)  
    write(lib4)  
  
if __name__ == '__main__':  
    test()
```

Seleccione una:

- a.  
No es correcto. En Python no hay forma de hacer que dos variables de tipo registro contengan los mismos datos y se mantengan independientes la una de la otra.
- b.  
No es correcto. La invocación a `init()` para inicializar `lib4` (`init(lib4, lib2.codigo, lib2.titulo, lib2.autor)`) hará que `lib2` y `lib4` apunten al mismo registro y desde allí en adelante cualquier cambio en una de ellas cambiará también la otra
- c.  
No es correcto. En lugar de invocar a `init()` para inicializar `lib4` con los datos de `lib2`, debería haberse asignado directamente `lib2` en `lib4` (o sea: `lib4 = Libro()` y luego `lib4 = lib2`).
- d.  
Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Sí. Es correcto y el esquema mostrado funciona sin problemas.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de tres libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el siguiente script:

```
class Libro:  
    pass  
  
lib1 = Libro()  
lib1.codigo = 2345  
lib1.titulo = 'El Aleph'  
  
lib2 = Libro()  
lib2.codigo = 1267  
lib2.titulo = 'Rayuela'  
lib2.autor = 'Julio Cortázar'  
  
lib3 = Libro()  
lib3.isbn = 123456767  
lib3.nombre = 'El Tunel'  
lib3.precio = 145.56
```

¿Es correcto el script mostrado o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.

No es correcto. Las tres variables *lib1*, *lib2* y *lib3* se están definiendo con *campos de nombres o identificadores diferentes* y eso no es posible.

b.

No es correcto. Las tres variables *lib1*, *lib2* y *lib3* se están definiendo con *distinta cantidad de campos* y eso no es posible.

c.

Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

d.

No es correcto. Las tres variables *lib1*, *lib2* y *lib3* pueden tener campos con nombres diferentes o incluso distinta cantidad de campos, pero al menos uno de los campos debe ser el mismo tipo para todos los registros.

¡Correcto!

La respuesta correcta es:

Sí. Es correcto y el esquema mostrado funciona sin problemas.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar datos de tres libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el siguiente script:

```
class Libro:  
    pass  
  
    def init(libro, cod, nom, aut):  
        libro.codigo = cod  
        libro.titulo = nom  
        libro.autor = aut  
  
    def write(libro):  
        print('Datos del libro:')  
        print('Código:', libro.codigo, ' - Título:', libro.titulo, ' - Autor:', libro.autor, ' - Precio:', libro.precio)  
  
    def test():  
  
        lib1 = Libro()  
        init(lib1, 2345, 'El Aleph', 'Jorge Luis Borges')  
  
        lib2 = Libro()  
        init(lib2, 1267, 'Rayuela', 'Julio Cortázar')  
  
        lib3 = Libro()  
        init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')  
        lib3.precio = 156.45  
  
        write(lib1)  
        write(lib2)  
        write(lib3)  
  
    if __name__ == '__main__':  
        test()
```

¿Es correcto el script mostrado o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.

Sí. Es correcto y el esquema mostrado funciona sin problemas.

b.

No es correcto. Luego de ser definida la variable *lib3*, se le está agregando el campo *precio* por separado y eso no es posible. El programa lanzará un error de intérprete en la linea *lib3.precio = 156.45*.

c.

No es correcto. Si se invoca a la función *init()* para inicializar los registros, entonces no debe invocarse a la función constructora *Libro()* (*init()* hace todo el trabajo).

d.

No es correcto. La función *write()* asume que el libro que toma como parámetro tiene siempre el campo *precio*, pero eso no es cierto al menos para las variables *lib1* y *lib2*. El programa lanzará un error de intérprete al intentar mostrar *lib1* o *lib2*. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

No es correcto. La función *write()* asume que el libro que toma como parámetro tiene siempre el campo *precio*, pero eso no es cierto al menos para las variables *lib1* y *lib2*. El programa lanzará un error de intérprete al intentar mostrar *lib1* o *lib2*.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. Suponga además que una vez creadas las variables *lib1*, *lib2* y *lib3* se quiere crear una nueva variable *lib4* que contenga los mismos datos que *lib2*, pero de tal forma que ambas se mantengan independientes: al modificar cualquier dato en una de las dos, no debe cambiar nada en la otra. ¿Hay algún problema con el programa siguiente, en base a estos requerimientos?

```
class Libro:  
    pass  
  
def init(libro, cod, nom, aut):  
    libro.codigo = cod  
    libro.titulo = nom  
    libro.autor = aut  
  
def write(libro):  
    print('Datos del libro:')  
    print('Código:', libro.codigo, ' - Título:', libro.titulo, ' - Autor:', libro.autor)  
  
def test():  
  
    lib1 = Libro()  
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borges')  
  
    lib2 = Libro()  
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')  
  
    lib3 = Libro()  
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')  
  
    write(lib1)  
    write(lib2)  
    write(lib3)  
  
    lib4 = lib2  
    lib4.autor = 'Adolfo Bioy Casares'  
  
    write(lib2)  
    write(lib4)  
  
if __name__ == '__main__':  
    test()
```

Seleccione una:

- a.

No es correcto. La asignación `lib4 = lib2` hace que ambas variables queden referenciando al mismo registro. Cualquier cambio que se haga de allí en más en una de las dos, cambiará entonces también a la otra.



¡Correcto!

- b.  
Sí. Es correcto y el esquema mostrado funciona sin problemas.

- c.  
No es correcto. Antes de la asignación `lib4 = lib2` debería haberse creado la variable `lib4` con la función constructora `Libro()` (es decir: `lib4 = Libro()`) y sólo entonces asignar `lib2`.

- d.  
No es correcto. En Python no hay forma de hacer que dos variables de tipo registro contengan los mismos datos y se mantengan independientes la una de la otra.

¡Correcto!

La respuesta correcta es:

No es correcto. La asignación `lib4 = lib2` hace que ambas variables queden referenciando al mismo registro. Cualquier cambio que se haga de allí en más en una de las dos, cambiará entonces también a la otra.

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere representar una fecha en un programa, y se propone utilizar un esquema basado en el uso de un registro, como se muestra en el siguiente script:

```
class Fecha:  
    pass  
  
f = Fecha()  
f.day = 27  
f.month = 7  
f.year = 2015  
  
print('Fecha registrada: ', f.day, '/', f.month,  
      '/', f.year, sep='')
```

¿Es correcto este enfoque o existe algún inconveniente? Si hay algún problema, indique cuál.

Seleccione una:

a.

No es correcto. La declaración de la clase *Fecha* no es válida: los campos deben declararse obligatoriamente dentro de ella (no puede dejarse vacía usando la instrucción *pass*).

b.

No es correcto. Los elementos de un registro no pueden accederse usando el operador *punto*.

c.

No es correcto. Un registro es un conjunto de datos que deben ser de tipos diferentes, y en este esquema todos los campos del registro *f* son del mismo tipo: *int*.

d.

Sí. Es correcto y el esquema mostrado funciona sin problemas. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Sí. Es correcto y el esquema mostrado funciona sin problemas.

**Pregunta 11**

Correcta

Puntúa 1 sobre 1

Suponga que el tipo registro *Cliente* está convenientemente declarado. ¿Qué hace el siguiente script en Python?

```
n = 8  
v = n * [None]  
for i in range(n):  
    v[i] = Cliente()
```

Seleccione una:

a.

Lanza un error de intérprete: la expresión  $n * [None]$  no tiene sentido.

b.

Crea un arreglo de  $n = 8$  componentes con su casilleros valiendo *None*, y luego en cada uno de esos componentes asigna una referencia a un registro vacío del tipo *Cliente*. ✓

¡Correcto!

c.

Crea un arreglo de  $n = 8$  componentes, para cada uno de esos componentes asigna referencias a registros del tipo *Cliente*, y cada *Cliente* inicializa todos sus campos con el valor 8.

d.

Crea un arreglo de  $n = 8$  componentes inicialmente valiendo *None*, y deja a ese vector con esas componentes en *None*.

¡Correcto!

La respuesta correcta es:

Crea un arreglo de  $n = 8$  componentes con su casilleros valiendo *None*, y luego en cada uno de esos componentes asigna una referencia a un registro vacío del tipo *Cliente*.

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

Suponga que la clase *Insumo* está convenientemente declarada para representar los insumos que se usan en la fabricación de una pieza. Asuma que también se ha definido un vector *pieza* de referencias a registros de tipo *Insumo* y que ese vector fue creado con *n* casilleros valiendo *None*. El programa mostrado en la Ficha 18 (problema 48) prevé que el arreglo será cargado luego con registros de ese tipo *Insumo*, entrando en la opción 1 del menú. Más abajo mostramos una variante de la función *opcion3()* para calcular el monto total insumido para fabricar la pieza completa, la cual a su vez invoca a la función *total\_value()* (que se ha dejado sin cambios) ¿Qué puede decirse respecto de la forma en que afecta al programa este replanteo de la función *opcion3()*?

```
def total_value(pieza):
    tv = 0
    for insumo in pieza:
        monto = insumo.valor * insumo.cantidad
        tv += monto
    return tv

def opcion3(pieza):
    print('Monto total en insumos para la pieza:', total_value(pieza))
```

Seleccione una:

a.

Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función *opcion3()* invocará a *total\_value()* enviándole un vector con todos sus casilleros valiendo *None*, y el ciclo iterador simplemente dejará el valor *None* en el acumulador *tv*. La función terminará en ese caso retornando *None*.

b.

Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función *opcion3()* invocará a *total\_value()* enviándole un vector con todos sus casilleros valiendo *None*, y el ciclo iterador automáticamente cambiará el *None* de cada casillero por un registro de tipo *Insumo* inicializado con valores cero. Por lo tanto en ese caso la función retornará el valor final 0.

c.

El programa funcionará correctamente de todos modos. Si cada casillero del vector valiese *None*, el ciclo iterador de la función *total\_value()* terminará sin hacer nada y la función retornará 0.

d.

Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función `opcion3()` invocará a `total_value()` enviándole un vector con todos sus casilleros valiendo `None`, y el ciclo iterador provocará un error y se interrumpirá ya que no existe en ese caso ningún campo llamado `valor` o `cantidad`. El programa se interrumpirá con un mensaje de error.



¡Correcto!

¡Correcto!

La respuesta correcta es:

Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función `opcion3()` invocará a `total_value()` enviándole un vector con todos sus casilleros valiendo `None`, y el ciclo iterador provocará un error y se interrumpirá ya que no existe en ese caso ningún campo llamado `valor` o `cantidad`. El programa se interrumpirá con un mensaje de error.

**Pregunta 13**

Correcta

Puntúa 1 sobre 1

Suponga que el registro *Materia* está convenientemente declarado para representar las asignaturas de una carrera universitaria. Asuma que también se ha definido un vector *mat* de referencias a registros de tipo *Materia* y que ese vector fue creado con n casilleros valiendo *None* y cargado luego con registros de ese tipo *Materia*, pero no se tiene seguridad en cuanto a que todos los casilleros del vector hayan sido correctamente asignados con una referencia a un registro. ¿Qué hace la siguiente función, suponiendo que está declarada en el mismo módulo que el vector?

```
def controlar(mat):  
    n = len(mat)  
    for i in range(n):  
        if mat[i] == None:  
            return i  
    return -1
```

Seleccione una:

a.

Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna *None*. Si ninguna casilla es *None*, retorna el valor -1.

b.

Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna el índice de la última casilla que sea *None*. Si ninguna casilla es *null*, retorna el valor -1.

c.

Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna el índice de la primera casilla que sea *None*. Si ninguna casilla es *None*, retorna el valor -1. ✓

¡Correcto!

d.

Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna un vector que contiene los índices de todas las casillas de *mat* que valen *None*. Si ninguna casilla es *None*, retorna el valor -1.

¡Correcto!

La respuesta correcta es:

Si el vector *mat* contiene al menos una casilla valiendo *None*, retorna el índice de la primera casilla que sea *None*. Si ninguna casilla es *None*, retorna el valor -1.

**Pregunta 14**

Correcta

Puntúa 2 sobre 2

Analice el siguiente script en Python:

```
__author__ = 'Cátedra de AED'

class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut

def test():
    a = Libro(1, 'AAA', 'aaa')
    b = Libro(2, 'BBB', 'bbb')
    c = Libro(3, 'CCC', 'ccc')

    n = 4
    v = n * [None]
    v[0] = a
    v[1] = b
    v[2] = c
    v[3] = v[1]

    a = None
    b = None
    c = None

    for i in range(n-1):
        v[i] = None

    print('Terminado...')

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

¿Cuál de las siguientes es correcta antes de llegar al `print()` final de la función `test()`?

Seleccione una:

a.

Los registros inicialmente apuntados por `a` y `c` quedan des-referenciados y son eliminados por el garbage collector. ✓

¡Correcto!

b.

Ninguno de los registros inicialmente apuntados por `a`, `b` y `c` queda des-referenciado.

c.

Sólo el registro inicialmente apuntado por `b` queda des-referenciado y es eliminado por el garbage collector.

d.

Todos los registros inicialmente apuntados por *a*, *b* y *c* quedan des-referenciados y son eliminados por el garbage collector.

¡Correcto!

La respuesta correcta es:

Los registros inicialmente apuntados por *a* y *c* quedan des-referenciados y son eliminados por el garbage collector.

**Pregunta 15**

Correcta

Puntúa 1 sobre 1

El siguiente módulo contiene la declaración de una clase *Estudiante*, incluyendo la función constructora `__init__()` y la función `to_string()` para inicializar y convertir a cadena un registro. Analice la forma en que está planteada la función `to_string()`... ¿Qué efecto provocarán en la cadena retornada por ese método los *campos de reemplazo* usados al invocar al método `format()`?

```
__author__ = 'Cátedra de AED'

class Estudiante:
    def __init__(self, leg, nom, prom):
        self.legajo = leg
        self.nombre = nom
        self.promedio = prom

    def to_string(estudiante):
        r = ''
        r += '{:<15}'.format('Legajo: ' + str(estudiante.legajo))
        r += '{:^30}'.format('Nombre: ' + estudiante.nombre)
        r += '{:>18}'.format('Promedio: ' + str(estudiante.promedio))
        return r
```

Seleccione una:

- a.  
El valor del campo *legajo* sale centrado, el *nombre* también sale centrado y el *promedio* se justifica la izquierda.
- b.  
El valor del campo *legajo* se justifica hacia la izquierda, el *nombre* sale centrado y el *promedio* se justifica también a la izquierda.
- c.  
El valor del campo *legajo* se justifica hacia la derecha, el *nombre* sale centrado y el *promedio* se justifica la izquierda.
- d.  
El valor del campo *legajo* se justifica hacia la izquierda, el *nombre* sale centrado y el *promedio* se justifica la derecha. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

El valor del campo *legajo* se justifica hacia la izquierda, el *nombre* sale centrado y el *promedio* se justifica la derecha.

**Pregunta 16**

Correcta

Puntúa 1 sobre 1

Analice el siguiente programa en Python:

```
__author__ = 'Cátedra de AED'

class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut

def test():
    n = 10
    lib = n * [None]
    for i in range(n):
        lib[i].isbn = i
    print('Terminado...')

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

¿Qué tiene de malo este programa?

Seleccione una:

 a.

No se pueden crear arreglos que contengan registros en Python.

 b.El problema es que en ninguna parte se invocó a la función *init()*, que debería ser invocada obligatoriamente. c.No se han creado los registros para cada casillero del arreglo. Cada casillero *est[i]* está valiendo *None* y por lo tanto es incorrecta la asignación *est[i].isbn = i* (provocará un error de intérprete y se interrumpirá el programa). ✓

¡Correcto!

 d.

No hay nada de malo con ese segmento.

¡Correcto!

La respuesta correcta es:

No se han creado los registros para cada casillero del arreglo. Cada casillero *est[i]* está valiendo *None* y por lo tanto es incorrecta la asignación *est[i].isbn = i* (provocará un error de intérprete y se interrumpirá el programa).

**Pregunta 17**

Correcta

Puntúa 2 sobre 2

Suponga que el tipo registro *Materia* está convenientemente declarado para representar las asignaturas de una carrera universitaria. Suponga también que ese tipo registro contiene un campo *nivel* para almacenar el año de la carrera al que pertenece esa materia (será 1 si la materia es de primer año, 2 si es de segundo, etc.) y algunos campos más. Asuma que se ha declarado también un vector *mat* de referencias a registros de tipo *Materia* y que ese vector fue creado y cargado correctamente con registros de tipo *Materia*. ¿Qué hace la siguiente función, suponiendo que está declarada en el mismo módulo que el vector?

```
def procesar(mat):
    n = len(mat)
    nuevo = []
    for i in range(n):
        if mat[i].nivel > 2:
            nuevo.append(mat[i])
    return nuevo
```

Seleccione una:

a.

Crea un segundo vector que contendrá todas las materias que no sean de primero ni de segundo año, y retorna finalmente el nuevo vector. ✓

¡Correcto!

b.

Crea un segundo vector que contendrá todas las materias que no sean de primero ni de segundo año, elimina esas materias del vector original, y retorna finalmente el nuevo vector.

c.

Crea un segundo vector que contendrá todas las materias de primero y segundo año, y retorna finalmente el nuevo vector.

d.

Crea un segundo vector que contendrá todas las materias del vector original que estén desde la casilla 2 en adelante, y retorna finalmente el nuevo vector.

¡Correcto!

La respuesta correcta es:

Crea un segundo vector que contendrá todas las materias que no sean de primero ni de segundo año, y retorna finalmente el nuevo vector.

**Pregunta 18**

Correcta

Puntúa 1 sobre 1

Suponga que el tipo registro *Inmueble* está convenientemente declarado. Suponga también que ese tipo contiene un campo *codigo* para almacenar el código de identificación del inmueble, otro para almacenar su precio, y algunos campos más. Asuma que también se ha definido un vector *inm* de referencias a registros de tipo *Inmueble* y que ese vector fue creado y cargado correctamente con registros de tipo *Inmueble*. Se desea ordenar el arreglo en *orden creciente de códigos de identificación*. ¿Está bien planteada la siguiente función para lograr ese ordenamiento?

```
def ordenar(inm):  
    n = len(inm)  
    for i in range(n-1):  
        for j in range(i+1, n):  
            if inm[i].codigo > inm[j].codigo:  
                inm[i], inm[j] = inm[j], inm[i]
```

Seleccione una:

a.

No, ya que esa función en realidad está ordenando el vector pero por precios en lugar de hacerlo por códigos.

b.

Sí. La función está bien planteada. ✓

¡Correcto!

c.

No. La función efectivamente ordena por códigos, pero lo hace en forma decreciente en lugar de hacerlo en forma creciente.

d.

No. La función hace incorrectamente los intercambios de registros y deja mezclados todos los datos iniciales.

¡Correcto!

La respuesta correcta es:

Sí. La función está bien planteada.

**Pregunta 19**

Correcta

Puntúa 1 sobre 1

Suponga que la clase *Insumo* está convenientemente declarada para representar los insumos que se usan en la fabricación de una pieza (ver problema 40 - Ficha 16). Asuma que también se ha definido un vector *pieza* de referencias a registros de tipo *Insumo* y que ese vector fue convenientemente creado y cargado con datos de *n* registros de tipo *Insumo*. En el programa de la Ficha 18 se incluyó una función sort que ordenaba el arreglo de menor a mayor de acuerdo a los valores del campo *codigo* de cada registro. Mostramos más abajo esa función sort(), pero ligeramente modificada respecto de la versión original ¿Qué puede decirse respecto de esta nueva versión modificada? (Por razones de claridad, hemos transcripto también la definición original de la clase *Insumo*).

```
class Insumo:  
    def __init__(self, cod=1, nom='', pre=0.0,  
    cant=0):  
        self.codigo = cod  
        self.nombre = nom  
        self.valor = pre  
        self.cantidad = cant
```

```
def sort(pieza):  
    n = len(pieza)  
    for i in range(n-1):  
        for j in range(i+1, n):  
            if pieza[i].nombre > pieza[j].nomb  
re:  
                pieza[i], pieza[j] = pieza[j],  
                pieza[i]
```

Seleccione una:

a.

Así como está planteada, la función sigue ordenando el vector de menor a mayor de acuerdo a los valores del campo *codigo*. No hay diferencia con lo que hacía la función original

b.

Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los valores del campo *codigo*, lo hace de acuerdo a los valores del campo *valor*.

c.

Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los valores del campo *codigo*, lo hace de acuerdo a los valores del campo *nombre* (el vector se ordenará en forma alfabética según los nombres de los insumos). ✓

|Correcto!

d.

Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los valores del campo *codigo*, lo hace de acuerdo a los valores del campo *cantidad*.

¡Correcto!

La respuesta correcta es:

Así como está planteada, la función sigue ordenando el vector de menor a mayor, pero en lugar de hacerlo de acuerdo a los valores del campo *codigo*, lo hace de acuerdo a los valores del campo *nombre* (el vector se ordenará en forma alfabética según los nombres de los insumos).

**Pregunta 20**

Correcta

Puntúa 1 sobre 1

En el problema 51 de la Ficha 18 se introdujo la explicación de cómo generar en forma aleatoria un vector de registros. ¿Cuáles podrían ser razones válidas por las que sería útil generar datos en forma automática? (Más de una puede ser cierta... marque TODAS las que considere correctas)

Seleccione una o más de una:

- a.

Para aplicar técnicas de generación de valores aleatorios y selección de valores aleatoriamente de una secuencia: aun cuando en un programa real los datos se carguen desde el teclado o desde un archivo, la generación automática es un recurso válido de aprendizaje y prueba.



¡Correcto!

- b.

Para ganar tiempo (sobre todo en fase de prueba o cuando el programa es una simulación): la generación automática ahorra mucho tiempo si los datos reales a cargar son muchos y/o si constan de combinaciones complicadas de valores.

✓ ¡Correcto!

- c.

No hay una razón válida para hacerlo. Los datos siempre deben ser cargados por teclado o recuperados desde un archivo externo si el mismo existe.

- d. Para facilitar la prueba de un programa antes de dejarlo en su versión definitiva.

✓ ¡Correcto!

¡Correcto!

Las respuestas correctas son: Para facilitar la prueba de un programa antes de dejarlo en su versión definitiva.,

Para ganar tiempo (sobre todo en fase de prueba o cuando el programa es una simulación): la generación automática ahorra mucho tiempo si los datos reales a cargar son muchos y/o si constan de combinaciones complicadas de valores.,

Para aplicar técnicas de generación de valores aleatorios y selección de valores aleatoriamente de una secuencia: aun cuando en un programa real los datos se carguen desde el teclado o desde un archivo, la generación automática es un recurso válido de aprendizaje y prueba.

**Pregunta 21**

Correcta

Puntúa 2 sobre 2

En el problema 51 de la Ficha 18 se introdujo la explicación de cómo generar en forma aleatoria un vector de registros. Uno de los campos de ese registro debía contener el número de patente de un vehículo (en formato argentino de 1994). En el programa original se mostró una forma simple de hacerlo. Se indican ahora algunas otras posibles maneras de lograrlo. ¿Cuáles de las siguientes ideas efectivamente logran generar una patente argentina? (Más de una puede ser cierta... marque TODAS las que considere correctas)

Seleccione una o más de una:

 a.

```
import random

letras = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + r
andom.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos)) + random.choice(digitos) +
random.choice(digitos))

patente = p1 + p2
print(patente)
```

 b.

```
import random

letras = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + r
andom.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = random.choice(digitos) + random.choice(digitos) +
random.choice(digitos)

patente = p1 + p2
print(patente)
```

 c.

```
import random

letras = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos)) + str(random.choice(digitos)) + str(random.choice(digitos))

patente = p1 + p2
print(patente)
```

✓ ¡Correcto!

d.

```
import random

letras = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = '0123456789'
p2 = random.choice(digitos) + random.choice(digitos) + random.choice(digitos)

patente = p1 + p2
print(patente)
```

✓ ¡Correcto!

¡Correcto!

Las respuestas correctas son:

```
import random

letras = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = '0123456789'
p2 = random.choice(digitos) + random.choice(digitos) + random.choice(digitos)

patente = p1 + p2
print(patente)
```

```
import random

letras = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) + random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos)) + str(random.choice(digitos)) + str(random.choice(digitos))

patente = p1 + p2
print(patente)
```

**Comenzado el** lunes, 17 de septiembre de 2018, 23:00

**Estado** Finalizado

**Finalizado en** lunes, 17 de septiembre de 2018, 23:16

**Tiempo empleado** 16 minutos 8 segundos

**Puntos** 13/13

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál es la diferencia entre la abstracción de datos y la abstracción funcional?

Seleccione una:

a.

La abstracción de datos busca captar el conjunto de datos más relevante para representar un tipo abstracto, mientras que la funcional buscar determinar el conjunto de procesos relevante para esos datos.



¡Correcto!

b.

La abstracción de datos busca captar el conjunto de procesos relevante para el tipo abstracto que se quiere implementar, mientras que la funcional buscar determinar el conjunto de datos más relevante para implementar ese tipo.

c.

Ninguna. Son sólo dos formas de referirse al mecanismo de abstracción.

d.

No existe un mecanismo de abstracción de datos ni un mecanismo de abstracción funcional. Existe sólo un mecanismo de abstracción, sin dividir en abstracción de datos y abstracción funcional.

¡Correcto!

La respuesta correcta es:

La abstracción de datos busca captar el conjunto de datos más relevante para representar un tipo abstracto, mientras que la funcional buscar determinar el conjunto de procesos relevante para esos datos.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere implementar un nuevo tipo de datos abstracto llamado *Fecha*, para permitir la manipulación de fechas en un programa en Python para gestión de eventos sociales (casamientos, fiestas de cumpleños, reuniones de egresados, etc.) ¿Cuál de las siguientes estrategias de abstracción sería la más adecuada?

Seleccione una:

- a.

*Abstracción de datos:* un registro con campos para el año, el mes, el día, el nombre del cliente, el tipo de evento, el monto presupuestado y una proyección metereológica para ese día. *Abstracción funcional:* funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento.

- b.

*Abstracción de datos:* un registro con tres campos para el año, el mes y el día. *Abstracción funcional:* funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento. ✓

¡Correcto!

- c.

*Abstracción de datos:* una tupla con tres componentes para el año, el mes y el día. *Abstracción funcional:* funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento.

- d.

*Abstracción de datos:* una variable de tipo *cadena de caracteres* para representar toda la fecha (por ejemplo: '2015/08/31'). *Abstracción funcional:* funciones para mostrar la fecha en distintos colores, imprimir la fecha en forma de cartel anunciador, cambiar el número del mes por el nombre del mes.

¡Correcto!

La respuesta correcta es:

*Abstracción de datos:* un registro con tres campos para el año, el mes y el día. *Abstracción funcional:* funciones para buscar una fecha dada, reservar una fecha para un evento, liberar una fecha para un evento y modificar los datos de una fecha para un evento.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expresiones describe la forma de trabajo general de una *Pila*?

Seleccione una:

a.  
FIFO (First In - First Out)

b.  
LIFO (Last In - First Out) ✓

¡Correcto!

c.  
OIFO (Ordered In - First Out)

d.  
OIRO (Ordered In - Random Out)

¡Correcto!

La respuesta correcta es:

LIFO (Last In - First Out)

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones de programación es la principal aplicación de una *Pila*?

Seleccione una:

- a.  
Procesar una secuencia de datos en el mismo orden en el que ingresaron.
- b.  
Procesar una secuencia de datos en orden aleatorio.
- c.  
Procesar una secuencia de datos en orden de menor a mayor.
- d.  
Procesar una secuencia de datos en orden inverso al de su entrada.



¡Correcto!

¡Correcto!

La respuesta correcta es:

Procesar una secuencia de datos en orden inverso al de su entrada.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

Suponga que se tiene una pila  $p$  con capacidad para almacenar números enteros y que las operaciones básicas `pop()`, `peek()` y `push()` están correctamente implementadas en el módulo `stack.py` presentado en clases. Suponga que se han insertado los siguientes valores: [8 - 6 - 5 - 3 - 7] (el número 8 es el valor del frente o tope de la Pila). ¿Cuál de las siguientes secuencias de instrucciones permite retirar el valor 5 de la pila, pero dejando el 6 y 8 nuevamente arriba? (es decir: ¿cuál de las siguientes secuencias, dejaría la pila  $p$  en el estado [8 - 6 - 3 - 7]?)

Seleccione una:

a.

```
n1 = stack.pop(p)  
n2 = stack.pop(p)  
x = stack.peek(p)  
stack.push(p, n2)  
stack.push(p, n1)
```

b.

```
n1 = stack.pop(p)  
n2 = stack.pop(p)  
x = stack.pop(p)  
stack.push(p, n2)  
stack.push(p, n1) ✓
```

¡Correcto!

c.

```
n1 = stack.pop(p)  
n2 = stack.pop(p)  
x = stack.pop(p)  
stack.push(p, n1)  
stack.push(p, n2)
```

d.

```
n1 = stack.pop(p)  
n2 = stack.pop(p)  
x = stack.pop(p)
```

¡Correcto!

La respuesta correcta es:

```
n1 = stack.pop(p)  
n2 = stack.pop(p)  
x = stack.pop(p)  
stack.push(p, n2)  
stack.push(p, n1)
```

**Pregunta 6**

Correcta

Puntúa 2 sobre 2

Suponga que se tiene una pila  $p$  en la cual se almacenaron ya una cierta cantidad de datos. ¿Cuál de las siguientes secuencias de instrucciones permite *invertir* la pila? (o sea: si la pila original  $p$  era: [3 - 4 - 6 - 7] (con el 3 al frente) ¿cuál de las siguientes permitiría dejar la pila  $p$  en el estado [7 - 6 - 4 - 3] (con el 7 al frente)?) (Suponga que  $p2$  y  $p3$  también son pilas, inicialmente vacías y listas para usar)

Seleccione una:

a.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p, stack.pop(p2))
```

b.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
```

c.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
while not stack.is_empty(p3):
    stack.push(p2, stack.pop(p3))
```

d.

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
while not stack.is_empty(p3):
    stack.push(p, stack.pop(p3))
```

✓ ¡Correcto!

¡Correcto!

La respuesta correcta es:

```
while not stack.is_empty(p):
    stack.push(p2, stack.pop(p))
while not stack.is_empty(p2):
    stack.push(p3, stack.pop(p2))
while not stack.is_empty(p3):
    stack.push(p, stack.pop(p3))
```

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expresiones describe la forma de trabajo general de una *Cola*?

Seleccione una:

- a.  
FIFO (First In - First Out) ✓  
¡Correcto!
- b.  
OIRO (Ordered In - Random Out)
- c.  
OIIFO (Ordered In - First Out)
- d.  
LIFO (Last In - First Out)

¡Correcto!

La respuesta correcta es:

FIFO (First In - First Out)

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones de programación es la principal aplicación de una Cola?

Seleccione una:

- a.  
Procesar una secuencia de datos en orden inverso al de su entrada.
- b.  
Procesar una secuencia de datos en orden de menor a mayor.
- c.  
Procesar una secuencia de datos en el mismo orden en el que ingresaron. ✓  
¡Correcto!
- d.  
Procesar una secuencia de datos en orden aleatorio.

¡Correcto!

La respuesta correcta es:

Procesar una secuencia de datos en el mismo orden en el que ingresaron.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

Suponga que se tiene una cola *c* en la cual se almacenaron ya una cierta cantidad de datos. ¿Cuál de las siguientes secuencias de instrucciones permite *invertir* la cola? (o sea: si la cola original *c* era: [3 - 4 - 6 - 7] (con el 3 al frente), ¿cuál de las siguientes permitiría dejar la cola *c* en el estado [7 - 6 - 4 - 3] (con el 7 al frente)?)

Seleccione una:

a.

```
# suponga que c2 es OTRA cola vacía y lista para usar...
while not queue.is_empty(c):
    queue.add(c2, queue.remove(c))
while not queue.is_empty(c2):
    queue.add(c, queue.remove(c2))
```

b.

```
# suponga que p1 y p2 son dos PILAS inicialmente vacías, y listas para usar
while not queue.is_empty(c):
    stack.push(p1, queue.remove(c))
while not stack.is_empty(p1):
    stack.push(p2, stack.pop(p1))
while not stack.is_empty(p2):
    queue.add(c, stack.pop(p2))
```

c.

```
# suponga que c2 y c3 son dos colas inicialmente vacías, y listas para usar
while not queue.is_empty(c):
    queue.add(c2, queue.remove(c))
while not queue.is_empty(c2):
    queue.add(c3, queue.remove(c2))
while not queue.is_empty(c3):
    queue.add(c, queue.remove(c3))
```

d.

```
# suponga que p es una PILA vacía y lista para usar...
while not queue.is_empty(c):
    stack.push(p, queue.remove(c))
while not stack.is_empty(p):
    queue.add(c, stack.pop(p))
```

✓ ¡Correcto! Efectivamente... si desea la inversión de una secuencia, una pila le permitirá lograrlo...

¡Correcto!

La respuesta correcta es:

```
# suponga que p es una PILA vacía y lista para usar...
while not queue.is_empty(c):
    stack.push(p, queue.remove(c))
while not stack.is_empty(p):
    queue.add(c, stack.pop(p))
```

**Pregunta 10**

Correcta

Puntúa 1 sobre 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 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.

b.

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

c.

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

d.

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 ( $O(1)$ ).

¡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.

---

**Comenzado el** lunes, 1 de octubre de 2018, 22:31

**Estado** Finalizado

**Finalizado en** lunes, 1 de octubre de 2018, 23:07

**Tiempo empleado** 36 minutos 41 segundos

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Para cada uno de los algoritmos o procesos listados en la columna de la izquierda, seleccione la expresión de orden que mejor describe su tiempo de ejecución en el peor caso.

Dadas dos matrices de orden  $n \times n$ , obtener la matriz producto aplicando el método tradicional.

O( $n^3$ ) (léase: "orden  $n$  al cubo")



Dado un arreglo ya ordenado  $v$  con  $n$  componentes, buscar un valor  $x$  aplicando búsqueda binaria.

O(log( $n$ ))



Dado un arreglo  $v$  con  $n$  componentes, buscar un valor  $x$  aplicando búsqueda secuencial.

O( $n$ )



Dado un arreglo  $v$  con  $n$  componentes, ordenarlo de menor a mayor mediante el algoritmo de selección directa.

O( $n^2$ ) (léase: "orden  $n$  al cuadrado")



Dado un arreglo  $v$  con  $n$  componentes, acceder y cambiar el valor del casillero  $v[k]$  (con  $0 \leq k \leq n-1$ ).

O(1)



¡Correcto!

La respuesta correcta es:

Dadas dos matrices de orden  $n \times n$ , obtener la matriz producto aplicando el método tradicional. → O( $n^3$ ) (léase: "orden  $n$  al cubo"),

Dado un arreglo ya ordenado  $v$  con  $n$  componentes, buscar un valor  $x$  aplicando búsqueda binaria.

→ O(log( $n$ )),

Dado un arreglo  $v$  con  $n$  componentes, buscar un valor  $x$  aplicando búsqueda secuencial. → O( $n$ ),

Dado un arreglo  $v$  con  $n$  componentes, ordenarlo de menor a mayor mediante el algoritmo de selección directa.

→ O( $n^2$ ) (léase: "orden  $n$  al cuadrado"),

Dado un arreglo  $v$  con  $n$  componentes, acceder y cambiar el valor del casillero  $v[k]$  (con  $0 \leq k \leq n-1$ ). → O(1)

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuántas comparaciones en el **peor caso** obliga a hacer una búsqueda secuencial en una *lista ordenada* (o en un *arreglo ordenado*) que contenga  $n$  valores?

Seleccione una:

a.

Peor caso:  $O(n^2)$  comparaciones.

b.

Peor caso:  $O(n)$  comparaciones. ✓

¡Correcto! Aún estando ordenado el arreglo, en el peor caso una búsqueda secuencial deberá llegar hasta el final si el valor buscado no existe en ese arreglo (o existe y está muy atrás)

c.

Peor caso:  $O(1)$  comparaciones.

d.

Peor caso:  $O(\log(n))$  comparaciones.

¡Correcto!

La respuesta correcta es:

Peor caso:  $O(n)$  comparaciones.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuál es la diferencia entre el *peor caso* y el *caso promedio* en el análisis de algoritmos?

Seleccione una:

a.

El *peor caso* es la configuración de datos de entrada más desfavorable para el algoritmo, mientras que el *caso promedio* describe una configuración aleatoria de datos (no pensada ni para favorecer ni para desfavorecer al algoritmo). 

¡Correcto!

b.

El *peor caso* es la configuración de datos de entrada más favorable para el algoritmo, mientras que el *caso promedio* describe una configuración aleatoria de datos (no pensada ni para favorecer ni para desfavorecer al algoritmo)

c.

El *peor caso* es la configuración de datos de entrada más favorable para el algoritmo, mientras que el *caso promedio* describe una configuración de datos pensada para desfavorecer al algoritmo.

d.

El *peor caso* es la configuración de datos de entrada más desfavorable para el algoritmo, mientras que el *caso promedio* describe una configuración de datos pensada para favorecer al algoritmo.

¡Correcto!

La respuesta correcta es:

El *peor caso* es la configuración de datos de entrada más desfavorable para el algoritmo, mientras que el *caso promedio* describe una configuración aleatoria de datos (no pensada ni para favorecer ni para desfavorecer al algoritmo).

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuáles de los siguientes son *factores de eficiencia comunes* a considerar en el análisis de algoritmos? (Más de una respuesta puede ser válida... marque *todas* las que considere correctas).

Seleccione una o más de una:



a.

El tiempo de ejecución. ✓

¡Correcto!



b.

La complejidad aparente del código fuente. ✓

¡Correcto!



c.

El consumo de memoria. ✓

¡Correcto!



d.

La calidad aparente de la interfaz de usuario.

¡Correcto!

Las respuestas correctas son:

El tiempo de ejecución.,

El consumo de memoria.,

La complejidad aparente del código fuente.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Qué significa decir que un algoritmo dado tiene un tiempo de ejecución **O(1)**?

Seleccione una:

- a.  
El tiempo de ejecución siempre es de un segundo, sin importar la cantidad de datos.
- b.  
El tiempo de ejecución es lineal: si aumenta el número de datos, aumenta el tiempo en la misma proporción.
- c.  
El tiempo de ejecución es logarítmico: a medida que aumenta el número de datos, aumenta el tiempo pero en forma muy suave.
- d.  
El tiempo de ejecución es constante, sin importar la cantidad de datos.



¡Correcto!

¡Correcto!

La respuesta correcta es:

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

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Qué significa decir que un algoritmo dado tiene un tiempo de ejecución  $O(n^2)$ ?

Seleccione una:

a.

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

b.

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.

c.

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

d.

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. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

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.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

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

Seleccione una:

a.

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.

b.

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

¡Correcto!

c.

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.

d.

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

¡Correcto!

La respuesta correcta es:

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

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Qué significa decir que un algoritmo dado tiene un tiempo de ejecución

**O(log(n))?**

Seleccione una:

 a.

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

 b.

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

 c.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. d.

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!

¡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 9**

Correcta

Puntúa 1 sobre 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 algoritmos 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 10**

Correcta

Puntúa 1 sobre 1

¿Con qué nombre general se conoce en la *Teoría de la Complejidad* a un problema para el cual sólo se conocen algoritmos cuyo tiempo de ejecución es exponencial (o sea, problemas para los que todas las soluciones conocidas son algoritmos con tiempo  $O(2^n)$ )?

Seleccione una:

- a.  
Problemas Intratables ✓
- b.  
Problemas Inmanejables
- c.  
Problemas Imperdonables
- d.  
Problemas Irresolubles

¡Correcto!

La respuesta correcta es:

Problemas Intratables

**Comenzado el** lunes, 8 de octubre de 2018, 14:15

**Estado** Finalizado

**Finalizado en** lunes, 8 de octubre de 2018, 15:25

**Tiempo empleado** 1 hora 9 minutos

**Puntos** 14/14

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Se tiene un algoritmo que realiza cierta cantidad de procesos sobre un conjunto de  $n$  datos y un minucioso análisis matemático ha determinado que la cantidad de procesos que el algoritmo realiza en el peor caso viene descripto por la función  $f(n) = 3n^3 + 5n^2 + 2n^{1.5}$ . ¿Cuál de las siguientes expresiones representa mejor el orden del algoritmo para el peor caso?

Seleccione una:

a.  
 $O(3n^3 + 5n^2 + 2n^{1.5})$

b.  
 $O(n^{1.5})$

c.  
 $O(n^3 + n^2)$

d.  
 $O(n^3)$  ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:  
 $O(n^3)$

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

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  $n^2$ ).

```
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( $n$ ) ✓

```
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( $n^3$ ) ✓

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

O(1) ✓

¡Correcto!

La respuesta correcta es:

```
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( $n$ ),

```
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( $n^3$ ),

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

→ O(1)

**Pregunta 3**

Correcta

Puntúa 2 sobre 2

Analice el siguiente esquema de una función en Python:

```
def procesar(n, m):
    for i in range(n+1):
        for j in range(m+1):
            # ... acciones sencillas a realizar...
            # ... suponga que no hay otro ciclo aquí...
            # ... y que sólo aparecen operaciones de tiempo constante...
```

¿Cuál de las siguientes expresiones de orden describe mejor el tiempo de ejecución de esta función en el peor caso?

Seleccione una:

a.  
 $O(n^*n)$

b.  
 $O(n*m)$  ✓

¡Correcto! Efectivamente, no hay problema alguno en que una expresión de orden se base en dos o más variables si el tamaño del problema depende de esas variables.

c.  
 $O(n)$

d.  
 $O(m)$

¡Correcto!

La respuesta correcta es:

$O(n*m)$

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuál es la **principal** característica de todos los métodos de ordenamiento conocidos como métodos simples o directos?

Seleccione una:

a.

Son muy fáciles de programar.

b.

Son muy veloces para cualquier tamaño del arreglo a ordenar.

c.

Tienen un tiempo de ejecución de orden cuadrático. 

¡Correcto!

d.

Tienen un tiempo de ejecución de orden lineal.

¡Correcto!

La respuesta correcta es:

Tienen un tiempo de ejecución de orden cuadrático.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características **correctas** del algoritmo *Shellsort*? (Más de una puede ser cierta... marque TODAS las que considere válidas)

Seleccione una o más de una:

- a.

En el caso promedio, el algoritmo *Shellsort* es tan eficiente como el *Heapsort* o el *Quicksort*, con tiempo de ejecución  $O(n \log(n))$ .

- b.

El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Selección Directa*, consistente en buscar iterativamente el menor (o el mayor) entre los elementos que quedan en el vector, para llevarlo a su posición correcta, pero de forma que la búsqueda del menor en cada vuelta se haga en tiempo logarítmico.

- c.

El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Inserción Directa* (o *Inserción Simple*), consistente en armar suconjuntos ordenados con elementos a distancia  $h > 1$  en las primeras fases, y terminar con  $h = 1$  en la última. ✓

¡Correcto!

- d.

El algoritmo *Shellsort* es complejo de analizar para determinar su rendimientos en forma matemática. Se sabe que para la serie de incrementos decrecientes usada en la implementación vista en las clases de la asignatura, tiene un tiempo de ejecución para el peor caso de  $O(n^{1.5})$ . ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Inserción Directa* (o *Inserción Simple*), consistente en armar suconjuntos ordenados con elementos a distancia  $h > 1$  en las primeras fases, y terminar con  $h = 1$  en la última.,

El algoritmo *Shellsort* es complejo de analizar para determinar su rendimientos en forma matemática. Se sabe que para la serie de incrementos decrecientes usada en la implementación vista en las clases de la asignatura, tiene un tiempo de ejecución para el peor caso de  $O(n^{1.5})$ .

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Qué diferencia existe entre el *conteo exhaustivo* de operaciones críticas y el *análisis asintótico* del comportamiento de una función en el análisis de algoritmos?

Seleccione una:

- a.

Ninguna. Ambas se refieren a la misma técnica básica del análisis de algoritmos

- b.

El *conteo exhaustivo* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones críticas que lleva a cabo un algoritmo, mientras que el *análisis asintótico* busca determinar el comportamiento general de una función para valores muy grandes del tamaño del problema. ✓

¡Correcto!

- c.

El *análisis asintótico* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones críticas que lleva a cabo un algoritmo, mientras que el *conteo exhaustivo* busca determinar el comportamiento general de una función para valores muy grandes del tamaño del problema.

- d.

El *conteo exhaustivo* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones **críticas** que lleva a cabo un algoritmo, mientras que el *análisis asintótico* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones **no críticas** que lleva a cabo un algoritmo.

¡Correcto!

Las respuestas correctas son:

El *conteo exhaustivo* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones críticas que lleva a cabo un algoritmo, mientras que el *análisis asintótico* busca determinar el comportamiento general de una función para valores muy grandes del tamaño del problema.,

El *conteo exhaustivo* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones **críticas** que lleva a cabo un algoritmo, mientras que el *análisis asintótico* busca determinar una expresión o fórmula que exprese de manera rigurosa la cantidad de operaciones **no críticas** que lleva a cabo un algoritmo.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

¿Qué se entiende, en el contexto del Análisis de Algoritmos, por un *Orden de Complejidad*?

Seleccione una:

- a.

Un conjunto o familia de funciones matemáticas que se comportan asintóticamente de la misma forma. ✓

¡Correcto!

- b.

Un conjunto o familia de algoritmos que resuelven el mismo problema.

- c.

Un conjunto o familia de subrutinas con similares objetivos (equivalente al concepto de *módulo*).

- d.

Un conjunto de datos ordenados.

¡Correcto!

Las respuestas correctas son:

Un conjunto o familia de funciones matemáticas que se comportan asintóticamente de la misma forma.,

Un conjunto o familia de algoritmos que resuelven el mismo problema.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Si los algoritmos de *ordenamiento simples* tienen todos un tiempo de ejecución  $O(n^2)$  en el peor caso, entonces: ¿cómo explica que las mediciones efectivas de los tiempos de ejecución de cada uno sean diferentes frente al mismo arreglo?

Seleccione una:

a.

La notación *Big O* rescata el término más significativo en la expresión que calcula el rendimiento, descartando constantes y otros términos que podrían no coincidir en los tres algoritmos. ✓

¡Correcto!

b.

Los tiempos deben coincidir. Si hay diferencias, se debe a errores en los instrumentos de medición o a un planteo incorrecto del proceso de medición.

c.

La notación *Big O* no se debe usar para estimar el comportamiento en el peor caso, sino sólo para el caso medio.

d.

La notación *Big O* no se usa para medir tiempos sino para contar comparaciones u otro elemento de interés. Es un error, entonces, decir que los tiempos tienen "*orden n cuadrado*".

¡Correcto!

La respuesta correcta es:

La notación *Big O* rescata el término más significativo en la expresión que calcula el rendimiento, descartando constantes y otros términos que podrían no coincidir en los tres algoritmos.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes *son correctas* en cuanto a los tiempos de ejecución de los algoritmos de ordenamiento clásicos? (Más de una puede ser cierta... marque TODAS las que considere válidas)

Seleccione una o más de una:

- a.

Algoritmo Quick Sort:  $O(n \cdot \log(n))$  en el caso promedio, pero  $O(n^2)$  en el peor caso. ✓

¡Correcto!

- b.

Algoritmo Heap Sort:  $O(n \cdot \log(n))$  tanto para el caso promedio como para el peor caso. ✓

¡Correcto!

- c.

Algoritmo Shell Sort:  $O(n^2)$  en el peor caso para la serie de incrementos decrecientes vista en clase.

- d.

Algoritmos directos o simples:  $O(n^2)$  en el peor caso para todos ellos. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Algoritmos directos o simples:  $O(n^2)$  en el peor caso para todos ellos.,

Algoritmo Quick Sort:  $O(n \cdot \log(n))$  en el caso promedio, pero  $O(n^2)$  en el peor caso.,

Algoritmo Heap Sort:  $O(n \cdot \log(n))$  tanto para el caso promedio como para el peor caso.

**Pregunta 10**

Correcta

Puntúa 3 sobre 3

Si se realiza un análisis preciso del ordenamiento por *Selección Directa* para un arreglo de  $n$  componentes, se llega a la conclusión que ese algoritmo hará  $n-1$  pasadas, con  $n-1$  comparaciones en la primera,  $n-2$  en la segunda, y así sucesivamente reduciendo de a 1 la cantidad de comparaciones hasta hacer sólo una comparación en la última pasada. Por lo tanto, el algoritmo hará *invariablemente* una cantidad total de  $\frac{1}{2}(n^2 - n)$  comparaciones. Sabiendo esto, ¿cuáles de las siguientes expresiones son correctas para describir la cantidad de comparaciones que hará el algoritmo, usando distintos tipos de notaciones? (Más de una respuesta puede ser correcta. Marque TODAS las que considere correctas)

Seleccione una o más de una:



a.

Cantidad de comparaciones:  $\Theta(n^2)$  ✓

¡Correcto!



b.

Cantidad de comparaciones:  $\Omega(n^2)$  ✓

¡Correcto!



c.

Cantidad de comparaciones:  $o(n^2)$ 

d.

Cantidad de comparaciones:  $O(n^2)$  ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Cantidad de comparaciones:  $O(n^2)$ ,Cantidad de comparaciones:  $\Omega(n^2)$ ,Cantidad de comparaciones:  $\Theta(n^2)$

**Comenzado el** viernes, 12 de octubre de 2018, 10:16

**Estado** Finalizado

**Finalizado en** viernes, 12 de octubre de 2018, 10:27

**Tiempo empleado** 10 minutos 34 segundos

**Puntos** 12/12

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **FALSA** respecto de la serialización en Python?

Seleccione una:

a.

La *serialización* es un proceso por el cual el contenido de una variable se convierte automáticamente en una secuencia de bytes listos para ser almacenados en un archivo, y luego recuperarse desde el archivo y volver a crear la variable original.

b.

En Python no se pueden serializar variables de tipo simple (variables de tipo int, float, bool, etc.) ✓

¡Correcto! efectivamente, esta es la afirmación falsa: en Python es perfectamente posible serializar valores de tipo simple y hemos mostrado ejemplos en el proyecto F[21] Archivos.

c.

El método del módulo *pickle* que permite grabar una variable directamente es *pickle.dump()* y el que permite recuperar una variable serializada es *pickle.load()*.

d.

El módulo *pickle* es uno de los que brindan funciones para serializar en Python, pero no es el único (existen otros, tales como *json*).

¡Correcto!

La respuesta correcta es:

En Python no se pueden serializar variables de tipo simple (variables de tipo int, float, bool, etc.)

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **CIERTA** respecto de la operación para obtener el tamaño en bytes de un archivo en Python?

Seleccione una:

- a.

La función `os.path.getsize()` toma como parámetro el nombre físico de un archivo, y retorna la longitud en bytes de ese archivo. ✓

¡Correcto!

- b.

La función `open()` toma como parámetro el nombre físico de un archivo y el modo de apertura deseado, abre el archivo y retorna el tamaño en bytes del mismo.

- c.

En Python no hay forma de obtener el tamaño en bytes de un archivo.

- d.

El método `tell()` de los objetos manejadores de archivos, no toma parámetro alguno y retorna siempre el tamaño en bytes del archivo para el cual fue invocado.

¡Correcto!

La respuesta correcta es:

La función `os.path.getsize()` toma como parámetro el nombre físico de un archivo, y retorna la longitud en bytes de ese archivo.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes es claramente falsa respecto de las características y propiedades de un objeto para manejar archivos en Python (un *file object*, tal como lo crea y lo retorna la función *open()*)?

Seleccione una:

a.

Un *file object* (o un *file-like object*) en última instancia permite interpretar el contenido de un archivo como si se tratase de un arreglo de bytes en memoria externa.

b.

Un *file object* (o un *file-like object*) representa sólo archivos de texto (y nunca archivos binarios). 

¡Correcto! efectivamente, esta afirmación es falsa: un file object represente un archivo. Da igual si luego de usa como archivo de texto o binario.

c.

Un *file object* (o un *file-like object*) representa archivos en los que es posible acceder en forma directa a cualquiera de sus bytes mediante el método *seek()*.

d.

Un *file object* (o un *file-like object*) contiene métodos que permiten tanto grabar como leer datos del archivo representado, independientemente de que eso también puede hacerse con funciones de serialización incluidas en módulos separados (como *pickle* o *json*).

¡Correcto!

La respuesta correcta es:

Un *file object* (o un *file-like object*) representa sólo archivos de texto (y nunca archivos binarios).

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes es **FALSA** respecto del *file pointer* en un *file object*?

Seleccione una:

 a.

La **única forma** en que puede cambiar el valor del *file pointer* de un archivo es invocando al método *seek()* del *file object* que lo maneja. ✓

¡Correcto! Efectivamente, esta afirmación es falsa. El valor del file pointer puede cambiar usar *seek()*, pero esa no es la única forma: su valor inicial es colocado por *open()* y luego también cambia de valor cada vez que realiza una lectura o una grabación.

 b.

El *file pointer* de un archivo puede apuntar a algún byte que esté fuera del archivo, *a la derecha del final del archivo*, pero no puede contener un valor negativo.

 c.

El valor inicial del *file pointer* es asignado por la función *open()* al abrir el archivo.

 d.

Cada vez que termina una operación de lectura o grabación en el archivo, el *file pointer* queda apuntando al byte siguiente al último que se leyó o grabó.

¡Correcto!

La respuesta correcta es:

La **única forma** en que puede cambiar el valor del *file pointer* de un archivo es invocando al método *seek()* del *file object* que lo maneja.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Si bien sabemos que todo archivo contiene datos representados en binario (y en ese sentido, todo archivo es un archivo binario), el hecho es que los archivos en general se clasifican en *archivos binarios* y *archivos de texto*. ¿Cuál es la diferencia entre ambos?

Seleccione una:

- a.

Todos los archivos son binarios en cuanto a su contenido. La distinción se hace en cuanto a cómo se *interpreta* ese contenido. En los archivos de texto, todos los bytes se interpretan como caracteres visualizables, salvo los saltos de línea y/o retornos de carro. Pero en los archivos binarios, cada byte puede representar cualquier tipo de dato. La interpretación depende del programa que procese a ese archivo. ✓

¡Correcto!

- b.

No hay ninguna diferencia. Todos los archivos son binarios, y esa distinción es incorrecta.

- c.

No es cierto que todos los archivos son binarios. Algunos contienen efectivamente texto, otros contienen imágenes, otros contienen números, y en definitiva pueden contener datos del tipo que sea.

- d.

Un archivo de texto sólo (en la plataforma *Windows*) debería contener bytes cuyos valores sean mayores a 31 (caracteres visualizables), y **sólo bytes mayores a 31**. La presencia de cualquier byte con valor menor a 32, lleva a que ese archivo sea considerado binario (por la presencia de bytes que no representan caracteres visualizables).

¡Correcto!

La respuesta correcta es:

Todos los archivos son binarios en cuanto a su contenido. La distinción se hace en cuanto a cómo se *interpreta* ese contenido. En los archivos de texto, todos los bytes se interpretan como caracteres visualizables, salvo los saltos de línea y/o retornos de carro. Pero en los archivos binarios, cada byte puede representar cualquier tipo de dato. La interpretación depende del programa que procese a ese archivo.

**Pregunta 6**

Correcta

Puntúa 2 sobre 2

Analice el siguiente script Python, en el cual se están grabando registros de tipo *Libro* en un archivo por serialización:

```
__author__ = 'Catedra de AED'

import pickle


class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut


def display(libro):
    print('ISBN:', libro.isbn, end='')
    print(' - Título:', libro.titulo, end='')
    print(' - Autor:', libro.autor)


def test():
    print('Prueba de grabación de varios registros...')
    lib1 = Libro(2134, 'Fundación', 'Isaac Asimov')
    lib2 = Libro(5587, 'Fundación e Imperio', 'Isaac Asimov')
    lib3 = Libro(3471, 'Segunda Fundación', 'Isaac Asimov')

    fd = 'libros.dat'
    m = open(fd, 'wb')
    pickle.dump(lib1, m)
    pickle.dump(lib2, m)
    m.close()
    print('Se grabaron varios registros en el archivo', fd)

    m = open(fd, 'rb')
    lib1 = pickle.load(m)
    lib2 = pickle.load(m)
    lib3 = pickle.load(m)
    m.close()

    print('Se recuperaron estos registros desde el archivo', fd, ':')
    display(lib1)
    display(lib2)
    display(lib3)


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

¿Hay algún problema con este script?

Seleccione una:

a.

Sí. El problema es que no se pueden grabar registros en un archivo si el mismo fue abierto con `open()`. Debió usar `open_register_file()`.

b.

Sí. El problema es que el archivo fue abierto en modo `wb` cuando se pretendió grabar, pero en ese modo no se puede grabar.

c.

No hay nada de malo en este segmento.

d.

Sí. Se grabaron sólo dos registros en el archivo, pero luego se intentaron tres lecturas. La tercera fallará por encontrar el final del archivo en forma inesperada. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Sí. Se grabaron sólo dos registros en el archivo, pero luego se intentaron tres lecturas. La tercera fallará por encontrar el final del archivo en forma inesperada.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

¿Qué ocurre si se intenta leer en un archivo y el *file pointer* del mismo está apuntando en ese momento al final del archivo (o sea, al primer byte ubicado fuera del archivo)?

Seleccione una:

- a.

No ocurre nada. La lectura no se realiza, pero tampoco se lanza un error. El programa simplemente ignora la orden y continúa.

- b.

Se leen los bytes que siguen al final del archivo, bajo responsabilidad del programador, y el programa continúa normalmente.

- c.

Se lanza un error de intérprete de la forma *EOFError*, y el programa se interrumpe. ✓

¡Correcto!

- d.

La pregunta no tiene sentido: el *file pointer* **no puede** estar apuntando al final del archivo.

¡Correcto!

La respuesta correcta es:

Se lanza un error de intérprete de la forma *EOFError*, y el programa se interrumpe.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Qué ocurre si en un programa Python se usa el método `seek()` de un *file object* y se salta con ese método a un byte que está más allá del final del archivo?

Seleccione una:

- a.

Se interrumpe el programa lanzando un error de la forma `EOFError`.

- b.

El método `seek()` no puede saltar a un byte que esté fuera del archivo, por lo tanto, no hará nada.

- c.

El *file pointer* del archivo quedará posicionado en ese byte, aunque esté fuera del archivo. ✓

¡Correcto!

- d.

El *file pointer* quedará posicionado en ese byte, y el tamaño del archivo se ajustará a ese byte.

¡Correcto!

La respuesta correcta es:

El *file pointer* del archivo quedará posicionado en ese byte, aunque esté fuera del archivo.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

Suponga que en un programa en Python necesita abrir un archivo de tal forma que:

- El archivo se maneje como archivo binario (y no como archivo de texto).
- Si el archivo no existía NO sea creado.
- Se permita tanto leer como grabar su contenido.
- No se destruya el contenido si el archivo ya existía.
- Se pueda leer en cualquier posición del archivo y también grabar en cualquier posición del mismo.

¿Cuál de los posibles modos de apertura disponibles para `open()` tendría que emplear al abrir ese archivo?

Seleccione una:

- a. El modo r+b. ✓ ¡Correcto!
- b. El modo r+t.
- c. El modo a+b.
- d. El modo rb.

¡Correcto!

La respuesta correcta es: El modo r+b.

**Pregunta 10**

Correcta

Puntúa 2 sobre 2

¿Qué valor tendrá el *file pointer* del archivo representado por *m* luego de terminar de realizar las operaciones que se indican? (ignore cualquier situación de error que podría producirse al abrir el archivo o al hacer las grabaciones: sólo analice las líneas dadas suponiendo un contexto correcto):

```
import pickle
import os.path

a = 25
b = 2.876
c = 'Hola mundo'

m = open('prueba.dat', 'wb')
pickle.dump(a, m) # suponer que aquí se grabaron 5 bytes...
pickle.dump(b, m) # suponer que aquí se grabaron 12 bytes...
pickle.dump(c, m) # suponer que aquí se grabaron 20 bytes...
m.close()
```

Seleccione una:

- a. 20
- b. 37 ✓ ¡Correcto!
- c. 32
- d. 5

¡Correcto!

La respuesta correcta es: 37

---

**Comenzado el** viernes, 12 de octubre de 2018, 10:29

**Estado** Finalizado

**Finalizado en** viernes, 12 de octubre de 2018, 12:28

**Tiempo empleado** 1 hora 58 minutos

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son **ciertas** en relación a un *programa de gestión ABM*? (Aclaración: más de una respuesta puede ser válida. Marque **todas** las que considere correctas)

Seleccione una o más de una:

- a.

Las operaciones básicas que un programa ABM permite realizar son las de altas, bajas y modificaciones de registros sobre un archivo. ✓

¡Correcto!

- b.

Un programa de gestión ABM ofrece un menú para que el operador pueda realizar operaciones de altas, bajas y modificaciones sobre un archivo, pero normalmente ofrece además opciones para otras operaciones como listados completos o filtrados, búsqueda de registros por diversos criterios, ordenamiento, etc. ✓

¡Correcto!

- c.

Un programa de gestión ABM ofrece un menú para que el operador pueda realizar exclusiva y únicamente operaciones de altas, bajas y modificaciones sobre un archivo, y justamente de allí proviene la designación de ABM (iniciales de las palabras altas, bajas, modificaciones).

- d.

Típicamente, en un programa ABM, la opción del menú para realizar bajas lanza un proceso de baja por marcado lógico, y las verdaderas bajas físicas se hacen con un proceso separado de depuración del archivo (que no suele estar disponible en el menú del usuario final, sino que se activa en forma automática en momentos no críticos del funcionamiento del programa) ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Las operaciones básicas que un programa ABM permite realizar son las de altas, bajas y modificaciones de registros sobre un archivo.,

Un programa de gestión ABM ofrece un menú para que el operador pueda realizar operaciones de altas, bajas y modificaciones sobre un archivo, pero normalmente ofrece además opciones para otras operaciones como listados completos o filtrados, búsqueda de registros por diversos criterios, ordenamiento, etc.,

Típicamente, en un programa ABM, la opción del menú para realizar bajas lanza un proceso de baja por marcado lógico, y las verdaderas bajas físicas se hacen con un proceso separado de depuración del archivo (que no suele estar disponible en el menú del usuario final, sino que se activa en forma automática en momentos no críticos del funcionamiento del programa)

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuál es la principal ventaja del *borrado físico* de un componente en un archivo?

Seleccione una:

- a.  
El borrado físico es más rápido que el marcado lógico
- b.  
El borrado físico evita la necesidad de ordenar el archivo para una búsqueda rápida
- c.  
**El archivo no ocupa más lugar que el necesario, por lo que optimiza el espacio utilizado.** ✓  
¡Correcto!
- d.  
El borrado físico garantiza que luego sean posibles los recorridos secuenciales en el archivo.

¡Correcto!

La respuesta correcta es:

El archivo no ocupa más lugar que el necesario, por lo que optimiza el espacio utilizado.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuál es la principal ventaja del *borrado por marcado lógico* en un archivo?

Seleccione una:

a.

El borrado por marcado lógico hace innecesario el borrado físico.

b.

El borrado por marcado lógico hace que el archivo ocupe sólo el espacio que necesita, optimizando el uso de la memoria externa.

c.

El borrado por marcado lógico no requiere agregar atributos o campos especiales en los registros que se graban en el archivo.

d.

El borrado por marcado lógico es más rápido que el físico, siempre que la búsqueda del componente a eliminar pueda hacerse velozmente. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

El borrado por marcado lógico es más rápido que el físico, siempre que la búsqueda del componente a eliminar pueda hacerse velozmente.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Si se desea implementar una aplicación ABM sobre un archivos de registros, *con la intención de poder hacer seeking y ubicar el file pointer en forma directa en el byte donde comienza cada registro*, ¿cuál de las siguientes opciones indica los principales problemas que se deben enfrentar?

Seleccione una:

- a. El manejo de registros que sólo contengan campos de tipo cadena de caracteres, pero todas con la misma cantidad de caracteres.
- b.  
El manejo de registros con campos de tipo cadena de caracteres de longitud variable, y en general, registros de tamaño no uniforme grabados en el archivo. ✓
- c.  
El manejo de registros que sólo contengan campos numéricos de tipo flotante.
- d.  
El manejo de registros de tamaño uniforme grabados en el archivo.

¡Correcto!

La respuesta correcta es:

El manejo de registros con campos de tipo cadena de caracteres de longitud variable, y en general, registros de tamaño no uniforme grabados en el archivo.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de alta de registros en un archivo*, *suponiendo que NO se admiten registros con clave repetida* en ese archivo?

Seleccione una:

a.

1) Abrir el archivo *m* en modo 'ab'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro *r* que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el archivo.

b.

1) Abrir el archivo *m* en modo 'a+b'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro *r* que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *eliminado* (*activo = False*), y grabar finalmente el registro en el archivo.

c.

1) Abrir el archivo *m* en modo 'a+b'. 2) Asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el archivo.

d.

1) Abrir el archivo *m* en modo 'a+b'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro *r* que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el arrchivo. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

1) Abrir el archivo *m* en modo 'a+b'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro *r* que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el arrchivo.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de baja lógica de registros en un archivo*?

Seleccione una:

a.

1) Abrir el archivo *m* en modo '*r+b*'. 2) Determinar si el archivo contiene o no a un registro con la misma clave que se quiere eliminar. 3) Si no existe un registro con esa clave, rechazar la baja. Si existe, asegurarse de marcar el registro como *no eliminado* (*activo = True*), y volver a grabar el registro en la misma posición que tenía en el archivo.

b.

1) Abrir el archivo *m* en modo '*rb*'. 2) Determinar si el archivo contiene o no a un registro con la misma clave que se quiere eliminar. 3) Si no existe un registro con esa clave, rechazar la baja. Si existe, asegurarse de marcar el registro como *eliminado* (*activo = False*), y volver a grabar el registro en la misma posición que tenía en el archivo.

c.

1) Abrir el archivo *m* en modo '*a+b*'. 2) Determinar si el archivo contiene o no a un registro con la misma clave que se quiere eliminar. 3) Si no existe un registro con esa clave, rechazar la baja. Si existe, asegurarse de marcar el registro como *eliminado* (*activo = False*), y volver a grabar el registro en la misma posición que tenía en el archivo.

d.

1) Abrir el archivo *m* en modo '*r+b*'. 2) Determinar si el archivo contiene o no a un registro con la misma clave que se quiere eliminar. 3) Si no existe un registro con esa clave, rechazar la baja. Si existe, asegurarse de marcar el registro como *eliminado* (*activo = False*), y volver a grabar el registro en la misma posición que tenía en el arrchivo. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

1) Abrir el archivo *m* en modo '*r+b*'. 2) Determinar si el archivo contiene o no a un registro con la misma clave que se quiere eliminar. 3) Si no existe un registro con esa clave, rechazar la baja. Si existe, asegurarse de marcar el registro como *eliminado* (*activo = False*), y volver a grabar el registro en la misma posición que tenía en el arrchivo.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de modificación de registros en un archivo*?

Seleccione una:

a.

1) Abrir el archivo *m* en modo 'ab'. 2) Determinar si el archivo contiene o no a un registro con la clave que se está buscando. 3) Si no existe un registro con esa clave, rechazar la modificación. Si existe, ofrecer al operador un menú para modificar los campos que desee. 4.) Asegurarse que el registro esté marcado como *no eliminado (activo = True)*, y volver a grabar el registro en la misma posición que tenía en el archivo.

b.

1) Abrir el archivo *m* en modo 'w+b'. 2) Determinar si el archivo contiene o no a un registro con la clave que se está buscando. 3) Si no existe un registro con esa clave, rechazar la modificación. Si existe, ofrecer al operador un menú para modificar los campos que desee. 4.) Asegurarse que el registro esté marcado como *no eliminado (activo = True)*, y volver a grabar el registro en la misma posición que tenía en el archivo.

c.

1) Abrir el archivo *m* en modo 'a+b'. 2) Determinar si el archivo contiene o no a un registro con la clave que se está buscando. 3) Si no existe un registro con esa clave, rechazar la modificación. Si existe, ofrecer al operador un menú para modificar los campos que desee. 4.) Asegurarse que el registro esté marcado como *eliminado (activo = False)*, y volver a grabar el registro en la misma posición que tenía en el archivo.

d.

1) Abrir el archivo *m* en modo 'r+b'. 2) Determinar si el archivo contiene o no a un registro con la clave que se está buscando. 3) Si no existe un registro con esa clave, rechazar la modificación. Si existe, ofrecer al operador un menú para modificar los campos que desee. 4.) Asegurarse que el registro esté marcado como *no eliminado (activo = True)*, y volver a grabar el registro en la misma posición que tenía en el archivo. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

1) Abrir el archivo *m* en modo 'r+b'. 2) Determinar si el archivo contiene o no a un registro con la clave que se está buscando. 3) Si no existe un registro con esa clave, rechazar la modificación. Si existe, ofrecer al operador un menú para modificar los campos que desee. 4.) Asegurarse que el registro esté marcado como *no eliminado (activo = True)*, y volver a grabar el registro en la misma posición que tenía en el archivo.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de depuración física de un archivo*, para eliminar efectivamente en ese proceso a todos los registros marcados como eliminados en ese archivo?

Seleccione una:

a.

1) Abrir el archivo *original* y el *temporal* en modo 'wb'. 2) Leer uno a uno los registros del *original*, y grabar en el *temporal* sólo aquellos que estén marcados como eliminados. 3) Eliminar el archivo *original* y cambiar el nombre del *temporal*. 4) Cerrar ambos archivos.

b.

1) Abrir el archivo *original* en modo 'rb' y abrir el archivo *temporal* en modo 'wb'. 2) Leer uno a uno los registros del *original*, y grabar en el *temporal* sólo aquellos que NO estén marcados como eliminados. 3) Cerrar ambos archivos. 4) Eliminar el archivo *original* y cambiar el nombre del *temporal*. ✓

¡Correcto!

c.

1) Abrir el archivo *original* y el *temporal* en modo 'wb'. 2) Leer uno a uno los registros del *original*, y grabar en el *temporal* sólo aquellos que NO estén marcados como eliminados. 3) Eliminar el archivo *original* y cambiar el nombre del *temporal*. 4) Cerrar ambos archivos.

d.

1) Abrir el archivo *original* y el *temporal* en modo 'wb'. 2) Leer uno a uno los registros del *original*, y grabar en el *temporal* sólo aquellos que NO estén marcados como eliminados. 3) Cerrar ambos archivos. 4) Eliminar el archivo *original* y cambiar el nombre del *temporal*.

¡Correcto!

La respuesta correcta es:

1) Abrir el archivo *original* en modo 'rb' y abrir el archivo *temporal* en modo 'wb'. 2) Leer uno a uno los registros del *original*, y grabar en el *temporal* sólo aquellos que NO estén marcados como eliminados. 3) Cerrar ambos archivos. 4) Eliminar el archivo *original* y cambiar el nombre del *temporal*.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un proceso de *listado completo de los registros de un archivo*, suponiendo que los registros *contienen un campo de marcado lógico* para gestionar eventuales bajas lógica?

Seleccione una:

a.

1) Abrir el archivo *m* en modo 'ab'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).

b.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como *no válido* (*activo = False*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).

c.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario no mostrar su contenido y detener el ciclo.

d.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido). ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de listado con filtro de los registros de un archivo*, suponiendo que los registros *contienen un campo de marcado lógico* para gestionar eventuales bajas lógica?

Seleccione una:

a.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*) y además cumple la condición de filtro, mostrar su contenido; en caso contrario no mostrar su contenido y detener el ciclo.

b.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*) y además cumple la condición de filtro, mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido). ✓

¡Correcto!

c.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).

d.

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como *no válido* (*activo = False*) y además cumple la condición de filtro, mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).

¡Correcto!

La respuesta correcta es:

1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*) y además cumple la condición de filtro, mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).

**Comenzado el** lunes, 29 de octubre de 2018, 17:38

**Estado** Finalizado

**Finalizado en** sábado, 3 de noviembre de 2018, 10:48

**Tiempo empleado** 4 días 17 horas

**Puntos** 12/12

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Para cada una de las funciones/métodos para manejo archivos en Python que se indican en la columna de la izquierda, seleccione la explicación que mejor describe su uso y aplicación:

- |                                 |   |
|---------------------------------|---|
| Método<br><i>seek()</i>         | Cambiar el valor del file pointer del archivo   |
| Función<br><i>open()</i>        | Abrir un archivo en un modo dado entre muchos posibles                                    |
| Función<br><i>pickle.load()</i> | Recuperar desde un archivo serializado el contenido de una variable/objeto.               |
| Función<br><i>pickle.dump()</i> | Grabar el contenido de una variable/objeto en un archivo mediante serialización           |
| Método<br><i>close()</i>        | Cerrar un archivo. De ser necesario, volcar primero el buffer de escritura en ese archivo |
| Método<br><i>flush()</i>        | Volcar el buffer de escritura de un archivo, sin cerrarlo                                 |
| Método <i>tell()</i>            | Consultar el valor del file pointer de un archivo   |

¡Correcto!

La respuesta correcta es:

Método *seek()* → Cambiar el valor del file pointer del archivo,

Función *open()* → Abrir un archivo en un modo dado entre muchos posibles,

Función *pickle.load()* → Recuperar desde un archivo serializado el contenido de una variable/objeto.,

Función *pickle.dump()* → Grabar el contenido de una variable/objeto en un archivo mediante serialización,

Método *close()* → Cerrar un archivo. De ser necesario, volcar primero el buffer de escritura en ese archivo,

Método *flush()* → Volcar el buffer de escritura de un archivo, sin cerrarlo, Método *tell()* → Consultar el valor del file pointer de un archivo

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

Para cada una de las descripciones de modos de apertura de un archivo que se indican en la columna de la izquierda, seleccione la cadena de caracteres usada en Python en la función `open()` para activar ese modo:

Archivo de texto. Lectura y grabación, ambas donde apunte el file pointer. El archivo debe existir.

`r+t`

Archivo de texto. Sólo lectura. El archivo debe existir.

`rt`

Archivo binario. Grabación al final, lectura en donde apunte el file pointer. El archivo se crea si no existe. Se preserva su contenido si existe.

`a+b`

Archivo binario. Sólo grabación, en donde apunte el file pointer. El archivo se crea si no existe. Se pierde su contenido si existe.

`wb`

Archivo binario. Sólo grabación al final. El archivo se crea si no existe. Se preserva si existe.

`ab`

Archivo binario. Sólo lectura. El archivo debe existir.

`rb`

¡Correcto!

La respuesta correcta es:

Archivo de texto. Lectura y grabación, ambas donde apunte el file pointer. El archivo debe existir.  
→ `r+t`,

Archivo de texto. Sólo lectura. El archivo debe existir. → `rt`,

Archivo binario. Grabación al final, lectura en donde apunte el file pointer. El archivo se crea si no existe. Se preserva su contenido si existe. → `a+b`,

Archivo binario. Sólo grabación, en donde apunte el file pointer. El archivo se crea si no existe. Se pierde su contenido si existe. → `wb`,

Archivo binario. Sólo grabación al final. El archivo se crea si no existe. Se preserva si existe. → `ab`,

Archivo binario. Sólo lectura. El archivo debe existir. → `rb`

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

Para cada una de las tres constantes del módulo `io` indicadas en la columna de la izquierda, seleccione su significado cuando se envían como segundo parámetro al método `seek()` de una variable `file object` que representa un archivo abierto.

`io.SEEK_SET` Suponer que el file pointer está ubicado al inicio del archivo

`✓`

`io.SEEK_CUR` Suponer que el file pointer está ubicado en su posición actual

`✓`

`io.SEEK_END` Suponer que el file pointer está ubicado al final del archivo

`✓`

¡Correcto!

La respuesta correcta es:

`io.SEEK_SET` → Suponer que el file pointer está ubicado al inicio del archivo,

`io.SEEK_CUR` → Suponer que el file pointer está ubicado en su posición actual,

`io.SEEK_END` → Suponer que el file pointer está ubicado al final del archivo

**Pregunta 4**

Correcta

Puntúa 1 sobre 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 25.

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

Apuntará al byte número 5.

`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).



¡Correcto!

La respuesta correcta es:

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

→ Apuntará al byte número 25.,

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

→ Apuntará al byte número 5.,

`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).

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

En la Ficha 24, problema número 51, se utilizó una función *buscar()* (que transcribimos más abajo) cuyo objetivo era determinar si alguno de los registros contenidos en un archivo *m* (tomado como parámetro), tenía su campo *dni* igual al valor *d* tomado también como parámetro. La función retornaba el número del byte en donde comenzaba el registro (si es que un registro con ese *dni* existía), o bien retornaba -1 (si el registro buscado no existía).

¿Cuáles de las siguientes afirmaciones son ciertas respecto del planteo de esta función? (Aclaración: más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

```
def buscar(m, d):
    global FD1
    t = os.path.getsize(FD1)

    fp_inicial = m.tell()
    m.seek(0, io.SEEK_SET)

    posicion = -1
    while m.tell() < t:
        fp = m.tell()
        vot = pickle.load(m)
        if vot.dni == d:
            posicion = fp
            break

    m.seek(fp_inicial, io.SEEK_SET)
    return posicion
```

Seleccione una o más de una:

 a.

La instrucción *m.seek(0, io.SEEK\_SET)* ubicada antes del ciclo de lectura está allí por razones de claridad, pero no es estrictamente necesaria. Su objetivo es llevar el file pointer al inicio del archivo antes de comenzar a leerlo, pero como se supone que el archivo viene abierto en un modo que permite lecturas entonces está garantizado que el file pointer estará ya ubicado al principio.

 b.

La instrucción *fp\_inicial = m.tell()* que se realiza antes del ciclo de lectura, se hace para poder recordar en qué posición estaba el file pointer del archivo *m* cuando *buscar()* fue invocada, de forma poder volver a posicionarlo allí con la intrucción *m.seek(fp\_inicial, io.SEEK\_SET)* antes de finalizar y dejar así el archivo tal como fue recibido. ✓

¡Correcto!

 c.

La instrucción *fp = m.tell()* que se realiza en el ciclo antes de leer un registro con *pickle.load()*, se hace para recordar en qué posición estaba el registro que se va a leer, de modo de poder retornarla luego si ese era el registro buscado. De otro modo, esa dirección se perdería pues *pickle.load()* mueve el file pointer al final del registro leído. ✓

¡Correcto!

 d.

La función *buscar()* recibe el archivo *m* como parámetro pero ni lo abre ni lo cierra: el supuesto es que el archivo *m* viene ya abierto, y en un modo que permite lecturas. Es responsabilidad del programador que invoca a la función *buscar()* cumplir con estos supuestos, ya que de lo contrario se producirá un error de runtime y el programa se interrumpirá cuando se intente leer el contenido de *m*. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

La instrucción `fp = m.tell()` que se realiza en el ciclo antes de leer un registro con `pickle.load()`, se hace para recordar en qué posición estaba el registro que se va a leer, de modo de poder retornarla luego si ese era el registro buscado. De otro modo, esa dirección se perdería pues `pickle.load()` mueve el file pointer al final del registro leído.,

La instrucción `fp_inicial = m.tell()` que se realiza antes del ciclo de lectura, se hace para poder recordar en qué posición estaba el file pointer del archivo `m` cuando `buscar()` fue invocada, de forma poder volver a posicionarlo allí con la intrucción `m.seek(fp_inicial, io.SEEK_SET)` antes de finalizar y dejar así el archivo tal como fue recibido.,

La función `buscar()` recibe el archivo `m` como parámetro pero ni lo abre ni lo cierra: el supuesto es que el archivo `m` viene ya abierto, y en un modo que permite lecturas. Es responsabilidad del programador que invoca a la función `buscar()` cumplir con estos supuestos, ya que de lo contrario se producirá un error de runtime y el programa se interrumpira cuando se intente leer el contenido de `m`.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

En la Ficha 24, problema número 52, se pidió gestionar un arreglo de registros que representaban artículos publicados en una revista científica (es decir, un arreglo con registros de tipo *Articulo*). Cada registro contenía un campo llamado *título* (para el título del artículo) y otro campo *codigo* (para el código numérico de identificación del artículo). El arreglo de registros debía cargarse por teclado, pero de forma que al agregar un nuevo artículo el arreglo quedase siempre ordenado de menor a mayor de acuerdo al valor del campo *título*. Entre las opciones a disponer en el menú, debía incluirse una para buscar un registro por su título, y otra para buscar un registro por su código numérico. La búsqueda por título se hacía con una función que aplicaba el algoritmo de búsqueda binaria, y la búsqueda por código se hacía con otra función que aplicaba búsqueda secuencial...

¿Por qué razón el programa propuesto incluyó dos funciones diferentes para las búsquedas pedidas? ¿Por qué fueron necesarios dos algoritmos diferentes para hacer las dos búsquedas requeridas en el enunciado?

Seleccione una:

a.

No hay ninguna razón para preferir un algoritmo sobre el otro, ya que ambos cumplen el objetivo pedido. El programa incluyó los dos por una razón didáctica, para mostrar al estudiante la aplicación de ambos en una misma unidad de análisis.

b.

En realidad, debería haberse aplicado el algoritmo de búsqueda binaria para ambos requerimientos de búsqueda. El arreglo está ordenado, por lo que la búsqueda binaria es aplicable.

c.

Porque el arreglo de artículos estaba ordenado en todo momento de acuerdo al valor del campo *título*, y no de acuerdo al campo *codigo*. Si se puede garantizar que el arreglo estará siempre ordenado por un campo en particular (*título* en este caso) y permanecerá así, no hay razón para no aplicar la búsqueda binaria cuando se busque un título. Pero como el arreglo no está ordenado por el campo *codigo*, y de acuerdo al contexto del enunciado, no queda entonces remedio y debe aplicarse la búsqueda secuencial cuando se pida buscar un código.



¡Correcto!

d.

En realidad, debería haberse aplicado el algoritmo de búsqueda secuencial para ambos requerimientos de búsqueda, sin importar si el arreglo está ordenado por algún campo en particular.

¡Correcto!

La respuesta correcta es:

Porque el arreglo de artículos estaba ordenado en todo momento de acuerdo al valor del campo *título*, y no de acuerdo al campo *codigo*. Si se puede garantizar que el arreglo estará siempre ordenado por un campo en particular (*título* en este caso) y permanecerá así, no hay razón para no aplicar la búsqueda binaria cuando se busque un título. Pero como el arreglo no está ordenado por el campo *codigo*, y de acuerdo al contexto del enunciado, no queda entonces remedio y debe aplicarse la búsqueda secuencial cuando se pida buscar un código.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

Suponga que el archivo *pacientes.med* ya existe y contiene varios registros del tipo registro *Paciente* grabados uno a uno por serialización en Python, mediante *pickle.dump()*. Suponga que el tipo *Paciente* es el mismo que el que se usó en la Ficha 23 con el problema número 49. Suponga ahora que se pide desarrollar un programa que use ese archivo para determinar la cantidad de días promedio que emplean entre todos los pacientes para regresar al consultorio (el promedio de los valores del campo fecha de los registros del archivo) y que una vez calculado ese promedio, se quiere mostrar por pantalla los datos de todos los pacientes que tengan el campo fecha mayor a ese promedio.

¿Es correcto el siguiente programa para cumplir con esa consigna? Si no lo es, ¿cuál es el problema?

```
__author__ = 'Catedra de AED'

import pickle
import os.path

class Paciente:
    def __init__(self, hc, nom, fec, cod):
        self.hist_clinica = hc
        self.nombre = nom
        self.fecha = fec
        self.cod_problema = cod

    def display(pac):
        print('Historia clínica:', pac.hist_clinica, end=' ')
        print('Nombre:', pac.nombre, end=' ')
        print('Días desde su última visita:', pac.fecha, end=' ')
        print('Código de enfermedad:', pac.cod_problema)

def mostrar_mayores():
    FD = 'pacientes.med'

    if not os.path.exists(FD):
        print('El archivo', FD, 'no existe...')
        print()
        return

    tbm = os.path.getsize(FD)
    m = open(FD, 'rb')

    ac, c = 0, 0
    while m.tell() < tbm:
        pac = pickle.load(m)
        ac += pac.fecha
        c += 1
    p = ac / c

    print('Pacientes con días de re-visita mayor al promedio de días:')
    while m.tell() < tbm:
        pac = pickle.load(m)
        if pac.fecha > p:
            display(pac)

    m.close()
    print()

# script principal...
if __name__ == '__main__':
    mostrar_mayores()
```

Seleccione una:

a.

Sí. El programa es cumple correctamente la consigna.

b.

No. No es correcto: Si se van a emplear dos ciclos para leer dos veces el contenido del archivo, entonces el archivo debería haber sido abierto en modo 'r+b' y no en modo 'rb'. Así como está planteado, el segundo ciclo provocará un error de runtime y se interrumpirá el programa.

c.

No. No es correcto: Cuando termina de recorrer el archivo la primera vez para calcular el promedio, el *file pointer* del archivo queda apuntando al final del mismo, por lo que el segundo ciclo no llega a activarse. No se mostrará nada en la pantalla (debería volver a 0 el file pointer antes del segundo ciclo, con `m.seek(0)`).

¡Correcto!

d.

No. No es correcto: Se está calculando mal el promedio, ya que la división debería hacerse con el operador de división entera (//) en lugar del operador de división real (/). Así como está planteado, el promedio daría un resultado *float*, y no se puede comparar valores *float* con valores *int*.

¡Correcto!

La respuesta correcta es:

No. No es correcto: Cuando termina de recorrer el archivo la primera vez para calcular el promedio, el *file pointer* del archivo queda apuntando al final del mismo, por lo que el segundo ciclo no llega a activarse. No se mostrará nada en la pantalla (debería volver a 0 el file pointer antes del segundo ciclo, con `m.seek(0)`).

#### Pregunta 8

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes instrucciones en Python insertará el valor de la variable `x` exactamente en la casilla número 4 del arreglo `v`, moviendo un casillero hacia la derecha a todos los elementos que estaban ya en `v` a partir de la casilla 4 (incluida) y sin perder a ninguno de ellos? (Suponga que el arreglo `v` tiene efectivamente al menos cuatro casilleros).

Seleccione una:

a.

`v[4] += x`

b.

`v[4:4] = [x]`

¡Correcto!

c.

`v[4] = x`

d.

`v[4] = [x]`

¡Correcto!

La respuesta correcta es:

`v[4:4] = [x]`

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

En la Ficha 24 se presentó el problema de agregar un registro a un arreglo en Python, suponiendo que el arreglo estaba ordenado de acuerdo a un campo de los registros que contiene, y con la premisa de que la inserción se haga de forma que el arreglo se mantenga ordenado luego de la inserción. Se presentaron dos versiones para la función `add_in_order()` que llevaba a cabo esa tarea: una basada en buscar el punto de inserción mediante búsqueda secuencial, y la otra mediante búsqueda binaria. ¿Cuáles de las siguientes afirmaciones son **ciertas** en relación a estas soluciones? (Aclaración: más de una respuesta puede ser válida. Marque todas las que considere correctas)

Seleccione una o más de una:

 a.

No debería aplicarse ninguna de las dos soluciones. Lo que debería hacerse es agregar cada nuevo registro al final del arreglo, y ordenar el arreglo luego de cada una de las inserciones.

 b.

En general, debería aplicarse la solución basada en búsqueda secuencial: es claramente más simple desde el punto de vista de la complejidad del código fuente, y las diferencias de velocidad de ejecución respecto del algoritmo de búsqueda binaria son siempre despreciables.

 c.

En general, debería aplicarse la solución basada en búsqueda binaria: es claramente más rápida ( $O(\log(n))$  en lugar de  $O(n)$ ) y el contexto del problema garantiza que es aplicable. ✓

¡Correcto! Esta es en realidad **LA** respuesta correcta... aunque algunas de las demás son admisibles en ciertos contextos.

 d.

La solución basada en búsqueda secuencial es más simple desde el punto de vista de la complejidad del código fuente, y podría admitirse su aplicación si se puede garantizar que el tamaño  $n$  del arreglo será realmente pequeño (unas pocas decenas de registros). ✓

¡Correcto! Aunque entienda: aceptamos que esto es admisible, pero en realidad el contexto del problema **exige** que se aplique búsqueda binaria... aunque sea para dejar el terreno preparado ante un eventual crecimiento inesperado del tamaño del arreglo.

¡Correcto!

Las respuestas correctas son:

En general, debería aplicarse la solución basada en búsqueda binaria: es claramente más rápida ( $O(\log(n))$  en lugar de  $O(n)$ ) y el contexto del problema garantiza que es aplicable.,

La solución basada en búsqueda secuencial es más simple desde el punto de vista de la complejidad del código fuente, y podría admitirse su aplicación si se puede garantizar que el tamaño  $n$  del arreglo será realmente pequeño (unas pocas decenas de registros).

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

Esta pregunta está pensada para cerrar el cuestionario, y agregar una pequeña nota de humor que permita al estudiante distenderse y aflojar un poco... lo cual viene bien estando tan cerca del final. Relájese un poco, ríase tanto como pueda, pero por favor, ¡asegúrese de responder correctamente! ¡Já Já!

Sabemos que la palabra *ABM* tan usada por los programadores de archivos viene de las iniciales de los procesos elementales que un programa *ABM* permite realizar ¿Cuáles son las palabras de las que provienen esas iniciales?

Seleccione una:

a.

ABM = Ancianos, Beodos, Motoqueros (A quien le calce el sayo... que no se note!!! Já Já!!)

b.

ABM = Ah!? Belgrano se Mueve? No parece... (Já Já!! Un saludo a los primos *Piratas*...)

c.

ABM = Altas, Bajas, Modificaciones (Humm... esta tiene cara de correcta...) ✓

¡Correcto!

d.

ABM = Antipáticas, Brujas, Mandonas... (No se enojen las damas... va con aprecio!!! Já Já!!)

¡Correcto!

La respuesta correcta es:

ABM = Altas, Bajas, Modificaciones (Humm... esta tiene cara de correcta...)

**Comenzado el** sábado, 3 de noviembre de 2018, 10:49

**Estado** Finalizado

**Finalizado en** sábado, 3 de noviembre de 2018, 11:00

**Tiempo empleado** 11 minutos

**Puntos** 12/12

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

Para cada uno de los métodos propios de un *file object* nombrados en la columna de la izquierda, seleccione en la columna de la derecha la descripción de su funcionamiento.

truncate(t)

Trunca (o expande) el contenido del archivo a una cantidad igual a t bytes. ▾



readable()

Retorna True si el archivo está disponible para ser leído. ▾



closed()

Retorna True si el archivo está cerrado. ▾



flush()

Graba en el archivo los buffers de grabación, sin cerrarlo. ▾



writelines(lines)

Graba el contenido de la lista "lines" en el archivo. ▾



writable()

Retorna True si el archivo está disponible para ser grabado. ▾



¡Correcto!

La respuesta correcta es: truncate(t) → Trunca (o expande) el contenido del archivo a una cantidad igual a t bytes., readable() → Retorna True si el archivo está disponible para ser leído., closed() → Retorna True si el archivo está cerrado., flush() → Graba en el archivo los buffers de grabación, sin cerrarlo., writelines(lines) → Graba el contenido de la lista "lines" en el archivo., writable() → Retorna True si el archivo está disponible para ser grabado.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Cuántos caracteres diferentes pueden representarse usando el estándar ASCII 7, y cuántos pueden representarse usando ASCII 8?

Seleccione una:

- a.  
256 con ASCII 7 y 128 con ASCII 8.
- b.  
32768 con ASCII 7 y 65536 con ASCII 8.
- c.  
4294967296 con ASCII 7 y 8589934592 con ASCII 8.
- d.  
128 con ASCII 7 y 256 con ASCII 8. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

128 con ASCII 7 y 256 con ASCII 8.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `read()`, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

 a.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, no se preservan: la cadena devuelta no contendrá saltos de línea de ningún tipo, en ninguna posición.

 b.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, se preservan y se retornan con la cadena devuelta. ✓

¡Correcto!

 c.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. ✓

¡Correcto!

 d.

Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación `cad = m.read(n)` lee una cantidad *n* de bytes del archivo los retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Si el archivo no llega a tener *n* bytes, se retorna la cadena vacía (""). ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*.

Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación `cad = m.read(n)` lee una cantidad *n* de bytes del archivo los retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Si el archivo no llega a tener *n* bytes, se retorna la cadena vacía ("").

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, se preservan y se retornan con la cadena devuelta.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `readline()`, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

a.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` lee *una sola línea de texto* del archivo y la retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. ✓

¡Correcto!

b.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` se detendrá en el primer carácter de salto de línea que `readline()` encuentre, o bien cuando encuentre el final del archivo. La cadena devuelta será asignada en este caso en la variable *cad*. ✓

¡Correcto!

c.

Si *m* es el *file object* que representa un archivo de texto, y la invocación `cad = m.readline()` no puede leer ninguna cadena (por la razón que sea...) se lanzará un error de runtime y el programa se interrumpirá.

d.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` se detendrá en el primer carácter de espacio en blanco que `readline()` encuentre, o bien cuando encuentre el final del archivo. La cadena devuelta será asignada en este caso en la variable *cad*.

¡Correcto!

Las respuestas correctas son:

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` lee *una sola línea de texto* del archivo y la retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` se detendrá en el primer carácter de salto de línea que `readline()` encuentre, o bien cuando encuentre el final del archivo. La cadena devuelta será asignada en este caso en la variable *cad*.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `readlines()` (con una `s` al final) contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

 a.

Si *m* es el *file object* que representa un archivo de texto, la invocación `v = m.readlines()` lee *el contenido completo* del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable *v*. *Ninguna* de las cadenas almacenadas en el arreglo *v* conservará el carácter de salto de línea que hubiese tenido en el archivo.

 b.

Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación `v = m.readlines(n)` lee *hasta n bytes del archivo* y los retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que haya logrado leer hasta completar *n* bytes, asignando ese arreglo en este caso en la variable *v*. ✓

¡Correcto!

 c.

Si *m* es el *file object* que representa un archivo de texto, la invocación `v = m.readlines()` lee *el contenido completo* del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable *v*. Cada una de las cadenas almacenadas en el arreglo *v* conservará el carácter de salto de línea que hubiese tenido en el archivo. ✓

¡Correcto!

 d.

Si *m* es el *file object* que representa un archivo de texto, la invocación `v = m.readlines()` lee *el contenido completo* del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable *v*.

✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Si *m* es el *file object* que representa un archivo de texto, la invocación `v = m.readlines()` lee *el contenido completo* del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable *v*.

Si *m* es el *file object* que representa un archivo de texto, la invocación `v = m.readlines()` lee *el contenido completo* del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable *v*. Cada una de las cadenas almacenadas en el arreglo *v* conservará el carácter de salto de línea que hubiese tenido en el archivo.,

Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación `v = m.readlines(n)` lee *hasta n bytes del archivo* y los retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por

cada línea de texto que haya logrado leer hasta completar  $n$  bytes, asignando ese arreglo en este caso en la variable  $v$ .

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `write()`, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:



a.

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  pero no agrega un carácter de salto de línea al final, a menos que  $cad$  ya lo contenga previamente. ✓

¡Correcto!



b.

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  y retorna la cantidad de caracteres que efectivamente grabó. ✓

¡Correcto!



c.

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  sin importar en qué modo haya sido abierto el archivo (sólo importa que se haya indicado que es de texto usando una 't' en el modo de apertura, y no una 'b').



d.

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  en forma directa y simple en un archivo de texto. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  en forma directa y simple en un archivo de texto.,

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  y retorna la cantidad de caracteres que efectivamente grabó.,

Si  $m$  es el *file object* que representa un archivo de texto y  $cad$  es una cadena de caracteres, la invocación  $m.write(cad)$  graba la cadena contenida en  $cad$  pero no agrega un carácter de salto de línea al final, a menos que  $cad$  ya lo contenga previamente.

**Pregunta 7**

Correcta

Puntúa 1 sobre 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 **Extended Binary Coded Decimal Interchange Code**) 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 8**

Correcta

Puntúa 2 sobre 2

¿Cuáles de las siguientes **son ciertas** en relación al procesamiento de archivos de texto que contienen números, representados como cadenas de caracteres a razón de un número/cadena por cada línea del archivo? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

 a.

Normalmente, cada línea de un archivo de este tipo contiene un "número" representado como una cadena de caracteres (sólo caracteres que representan dígitos) y cada número se separa del siguiente con un salto de línea u otro carácter clásico como un espacio en blanco, un tabulador, una coma, etc. ✓

¡Correcto!

 b.

Cuando se necesita procesar un archivo así, se leen desde el archivo las cadenas que representan números, se las convierte a números en el formato correcto (*int* o *float*), se almacenan esos números en alguna estructura de datos (como un arreglo), y finalmente se retorna esa estructura. ✓

¡Correcto!

 c.

El motivo por el que se suelen almacenar números como cadenas en un archivo de texto, es el de evitar problemas de incompatibilidad de formatos numéricos, si se prevé que distintos programadores usen lenguajes diferentes para procesar el mismo archivo. ✓

¡Correcto!

 d.

En realidad, no hay ningún motivo válido que justifique el uso de archivos con este criterio. Se deben almacenar directamente los números, en su formato numérico original, sin convertirlos a cadenas.

¡Correcto!

Las respuestas correctas son:

Normalmente, cada línea de un archivo de este tipo contiene un "número" representado como una cadena de caracteres (sólo caracteres que representan dígitos) y cada número se separa del siguiente con un salto de línea u otro carácter clásico como un espacio en blanco, un tabulador, una coma, etc.,

Cuando se necesita procesar un archivo así, se leen desde el archivo las cadenas que representan números, se las convierte a números en el formato correcto (*int* o *float*), se almacenan esos números en alguna estructura de datos (como un arreglo), y finalmente se retorna esa estructura.,

El motivo por el que se suelen almacenar números como cadenas en un archivo de texto, es el de evitar problemas de incompatibilidad de formatos numéricos, si se prevé que distintos programadores usen lenguajes diferentes para procesar el mismo archivo.

**Pregunta 9**

Correcta

Puntúa 1 sobre 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 carácter 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 distinguen las mayúsculas de las minúsculas dentro del archivo.

- d.

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.

¡Correcto!

La respuesta correcta es:

El tratamiento dado al *salto de linea*: en algunas plataformas se representa con un carácter 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 10**

Correcta

Puntúa 2 sobre 2

¿Cuál de las siguientes **es cierta** respecto del uso y aplicación del método `seek()` en archivos de texto con Python?

Seleccione una:

a.

Tanto da que el archivo sea de texto o binario, en Python el método `seek()` se aplica exactamente en la misma forma, sin restricciones.

b.

Si el archivo es de texto, en Python el método `seek()` sólo puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`). La única excepción es la de reponer el *file pointer* al final del archivo (`m.seek(0, io.SEEK_END)`) ✓

¡Correcto!

c.

Si el archivo es de texto, en Python el método `seek()` puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`) o desde la posición actual del file pointer (segundo parámetro igual a `io.SEEK_CUR = 1`), sin restricciones en ambos casos.

d.

Si el archivo es de texto, en Python el método `seek()` puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`) o desde el final del archivo (segundo parámetro igual a `io.SEEK_SET = 2`), sin restricciones en ambos casos.

¡Correcto!

La respuesta correcta es:

Si el archivo es de texto, en Python el método `seek()` sólo puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`). La única excepción es la de reponer el *file pointer* al final del archivo (`m.seek(0, io.SEEK_END)`)

---

**Comenzado el** martes, 6 de noviembre de 2018, 15:07

**Estado** Finalizado

**Finalizado en** miércoles, 7 de noviembre de 2018, 22:44

**Tiempo empleado** 1 día 7 horas

**Puntos** 15/15

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 2 sobre 2

El cálculo del valor  $a^n$  (para simplificar, asumimos  $a > 0$  y  $n \geq 0$  y sabiendo que si  $n = 0$  entonces  $a^0 = 1$ ) es igual a multiplicar  $n$  veces el número  $a$  por sí mismo. Por caso,  $5^3 = 5 * 5 * 5 = 125$ . Sabiendo esto, ¿cuál de las siguientes sería una *definición recursiva* matemáticamente correcta de la operación *potencia(a, n)*?

Seleccione una:

a.

$$\text{potencia}(a, n) \begin{cases} = 1 & \text{si } n == 0 \\ = a * \text{potencia}(a, n) & \text{si } n > 0 \\ (\text{a y n enteros,} \\ \text{a} > 0 \text{ y } n \geq 0) \end{cases}$$

b.

$$\text{potencia}(a, n) \begin{cases} = 1 & \text{si } n == 0 \\ = a * \text{potencia}(a-1, n) & \text{si } n > 0 \\ (\text{a y n enteros,} \\ \text{a} > 0 \text{ y } n \geq 0) \end{cases}$$

c.

$$\text{potencia}(a, n) \begin{cases} = 1 & \text{si } n == 0 \\ = a + \text{potencia}(a, n-1) & \text{si } n > 0 \\ (\text{a y n enteros,} \\ \text{a} > 0 \text{ y } n \geq 0) \end{cases}$$

d.

$$\text{potencia}(a, n) \begin{cases} = 1 & \text{si } n == 0 \\ = a * \text{potencia}(a, n-1) & \text{si } n > 0 \\ (\text{a y n enteros,} \\ \text{a} > 0 \text{ y } n \geq 0) \end{cases}$$



¡Correcto!

¡Correcto!

La respuesta correcta es:

$$\text{potencia}(a, n) \begin{cases} = 1 & \text{si } n == 0 \\ = a * \text{potencia}(a, n-1) & \text{si } n > 0 \\ (\text{a y n enteros,} \\ \text{a} > 0 \text{ y } n \geq 0) \end{cases}$$

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

Suponga que se quiere plantear una **definición recursiva** del concepto de **bosque**. ¿Cuál de las siguientes propuestas generales es correcta y constituye la mejor definición?

Seleccione una:

- a.

Un bosque es un conjunto de árboles que puede estar vacío, o puede contener uno o más árboles agrupados con otro bosque. ✓

¡Correcto!

- b.

Un bosque es un conjunto de árboles que puede estar vacío, o puede contener n árboles (con  $n > 0$ ).

- c.

Un bosque es un bosque.

- d.

Un bosque es un conjunto que puede contener uno o más árboles agrupados con otro bosque.

¡Correcto!

La respuesta correcta es:

Un bosque es un conjunto de árboles que puede estar vacío, o puede contener uno o más árboles agrupados con otro bosque.

**Pregunta 3**

Correcta

Puntúa 1 sobre 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 consumo de memoria. ✓

¡Correcto!



b.

La complejidad aparente del código fuente. ✓

¡Correcto!



c.

El diseño de la interfaz de usuario.



d.

El tiempo de ejecución. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

El tiempo de ejecución.,

El consumo de memoria.,

La complejidad aparente del código fuente.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Suponga que la versión recursiva de la función para el cálculo del factorial que se presentó en clases fuese redefinida en la forma siguiente:

```
def factorial(n):
    f = n * factorial(n - 1)
    if n == 0:
        return 1
    return f
```

¿Cuá es el problema (si lo hubiese) con esta versión de la función?

Seleccione una:

a.

Para calcular el factorial necesita un ciclo for que genere los números a multiplicar. Así planteada, la función hace un único primer cálculo incompleto y lo retorna.

b.

No hay ningún problema. Funciona correctamente.

c.

Contiene todos los elementos que debería tener una función recursiva bien planteada, pero en el orden incorrecto: la condición para chequear si  $n$  es cero debería estar antes que la llamada recursiva para evitar la recursión infinita. ✓

¡Correcto!

d.

La función siempre retorna un 1, sin importar cual sea el valor de  $n$ .

¡Correcto!

La respuesta correcta es:

Contiene todos los elementos que debería tener una función recursiva bien planteada, pero en el orden incorrecto: la condición para chequear si  $n$  es cero debería estar antes que la llamada recursiva para evitar la recursión infinita.

**Pregunta 5**

Correcta

Puntúa 2 sobre 2

A lo largo del curso hemos visto la forma de plantear programas controlados por menú de opciones, y sabemos que esos programas incluyen un ciclo para el control del proceso. Pero también podríamos hacer un planteo basado en recursión, sin usar el ciclo, en forma similar a la que se muestra aquí:

```
__author__ = 'Cátedra de AED'

def menu():
    print('1. Opcion 1')
    print('2. Opcion 2')
    print('3. Salir')
    op = int(input('Ingrese opcion: '))
    if op == 1:
        print('Eligio la opcion 1...')
    elif op == 2:
        print('Eligio la opcion 2...')
    elif op == 3:
        return
    menu()

# script principal...
menu()
```

¿Hay algún problema o contraindicación respecto del uso y aplicación de esta versión recursiva para la gestión de un menú? Seleccione la respuesta que mejor describa este punto.

Seleccione una:

a.

Esta versión recursiva posiblemente sea más simple y compacta que la versión basada en un ciclo, pero la versión recursiva pide memoria de stack cada vez que se invoca, por lo que es más conveniente la iterativa. ✓

¡Correcto! De todos modos, si la idea es usar el menú sólo para un par de operaciones y terminar de inmediato, no habrá problema alguno. Lo malo sería que el programa funcione en forma permanente, las 24 horas del día y todos los días... pues en ese caso, se corre el serio riesgo de provocar un overflow de stack.

b.

Esta versión recursiva es completamente equivalente a la versión iterativa. No hay ningún motivo para preferir la una sobre la otra. Da lo mismo si el programador usa la recursiva o la iterativa.

c.

La versión recursiva que se mostró no funciona. Sólo permite cargar una vez la opción elegida, e inmediatamente se detiene.

d.

No hay ningún problema ni contraindicación. Y no sólo eso: la versión recursiva es más simple y compacta que la versión basada en un ciclo, por lo cual es incluso preferible usarla.

¡Correcto!

La respuesta correcta es:

Esta versión recursiva posiblemente sea más simple y compacta que la versión basada en un ciclo, pero la versión recursiva pide memoria de stack cada vez que se invoca, por lo que es más conveniente la iterativa.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes era el nombre verdadero del matemático conocido como *Fibonacci*, que fue quien definió la famosa *Sucesión de Fibonacci*?



Seleccione una:

- a.  
Carl Gauss
- b.  
Leonardo Pisano ✓  
¡Correcto!
- c.  
Giuseppe Peano
- d.  
Leonhard Euler

¡Correcto!

La respuesta correcta es:

Leonardo Pisano

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

El producto de dos números  $a$  y  $b$  mayores o iguales cero, es en última instancia *una suma*: se puede ver como sumar  $b$  veces el número  $a$ , o como sumar  $a$  veces el número  $b$ . Por ejemplo:  $5 * 3 = 5 + 5 + 5$  o bien  $5 * 3 = 3 + 3 + 3 + 3 + 3$ . Sabiendo esto, se puede intentar hacer una definición recursiva de la operación  $\text{producto}(a, b)$  [ $a \geq 0, b \geq 0$ ], que podría ser la que sigue:

$$\text{producto}(a, b) = \begin{cases} 0 & \text{si } a == 0 \text{ o } b == 0 \\ a + \text{producto}(a, b-1) & \text{si } a > 0 \text{ y } b > 0 \end{cases}$$

[ $a$  y  $b$  enteros,  $a \geq 0$  y  $b \geq 0$ ]

¿Es correcto el siguiente planteo de la función  $\text{producto}(a, b)$ ?

```
def producto(a, b):
    if a == 0 or b == 0:
        return 0
    return a + producto(a, b-1)
```

Seleccione una:

- a.  
No. La propia definición previa del concepto de producto es incorrecta: un producto es una multiplicación, y no una suma...
- b.  
Sí. Es correcto. ✓  
¡Correcto!
- c.  
No. La última línea debería decir *return a + producto(a-1, b-1)* en lugar de *return a + producto(a, b-1)*.
- d.  
No. La función sugerida siempre retornará 0, sean cuales fuesen los valores de  $a$  y  $b$ .

¡Correcto!

La respuesta correcta es:

Sí. Es correcto.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

En la tabla siguiente se muestra un resumen comparativo entre las versiones iterativa (basada en ciclos) y recursiva del algoritmo de cálculo del término  $n$ -ésimo de la Sucesión de Fibonacci. Note que se han dejado sin completar los casilleros que corresponden al tiempo de ejecución para ambos casos.

Cálculo del término $n$ -ésimo de Fibonacci [F(n)]	Complejidad código fuente	Consumo de memoria	Tiempo de ejecución
Versión iterativa	Aceptable	Constante (no depende de $n$ )	????
Versión recursiva	Optima	Proporcional a $n$ (lineal)	????

¿Cuál es la estimación para el tiempo de ejecución de ambos algoritmos?

Seleccione una:

a.

Versión iterativa: tiempo proporcional a  $n$  (lineal) - Versión recursiva: tiempo proporcional a  $n$  (lineal)

b.

Versión iterativa: tiempo constante (no depende de  $n$ ) - Versión recursiva: tiempo proporcional a  $b^n$  (exponencial, para algún  $b > 1$ )

c.

Versión iterativa: tiempo proporcional a  $n$  (lineal) - Versión recursiva: tiempo proporcional a  $b^n$  (exponencial, para algún  $b > 1$ ) ✓

¡Correcto! Efectivamente... la cantidad de instancias recursivas que llegan a crearse a lo largo de todo el proceso recursivo está en el orden de  $1.8^n$  (lo cual es muy mala noticia...)

d.

Versión iterativa: tiempo proporcional a  $b^n$  (exponencial, para algún  $b > 1$ ) - Versión recursiva: tiempo proporcional a  $n$  (lineal)

¡Correcto!

La respuesta correcta es:

Versión iterativa: tiempo proporcional a  $n$  (lineal) - Versión recursiva: tiempo proporcional a  $b^n$  (exponencial, para algún  $b > 1$ )

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes propuestas generales son **ciertas** en relación al segmento de memoria conocido como *Stack Segment*?

Seleccione una o más de una:



a.

El Stack Segment funciona como una pila (o apilamiento) de datos (modalidad *LIFO*: Last In - First Out): el último dato en llegar, se almacena en la cima del stack, y será por eso el primero en ser retirado.



¡Correcto!



b.

El Stack Segment se utiliza como soporte interno *sólo* en el proceso de invocación a funciones recursivas: se va llenando a medida que la cascada recursiva se va desarrollando, y se vacía a medida que se produce el proceso de regreso o vuelta atrás.



c.

El Stack Segment funciona como una cola (o fila) de datos (modalidad *FIFO*: First In - First Out): el primer dato en llegar, se almacena al frente del stack, y será por eso el primero en ser retirado.



d.

El Stack Segment se utiliza como soporte interno en el proceso de invocación a funciones (*con o sin recursividad*): se va llenando a medida que se desarrolla la cascada de invocaciones, y se vacía a medida que se produce el proceso de regreso o vuelta atrás.



¡Correcto!

¡Correcto!

Las respuestas correctas son:

El Stack Segment funciona como una pila (o apilamiento) de datos (modalidad *LIFO*: Last In - First Out): el último dato en llegar, se almacena en la cima del stack, y será por eso el primero en ser retirado.,

El Stack Segment se utiliza como soporte interno en el proceso de invocación a funciones (*con o sin recursividad*): se va llenando a medida que se desarrolla la cascada de invocaciones, y se vacía a medida que se produce el proceso de regreso o vuelta atrás.

**Pregunta 10**

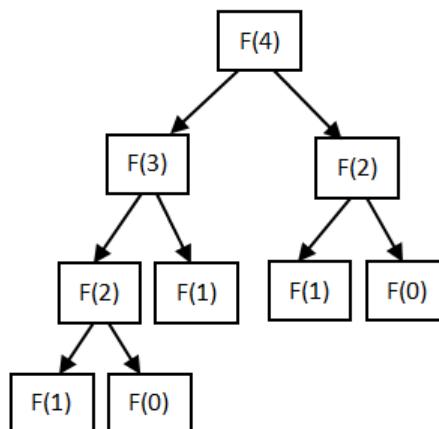
Correcta

Puntúa 2 sobre 2

Sabemos que la recursión es una de las técnicas o estrategias básicas para el planteo de algoritmos y que una de sus ventajas es que permite el diseño de algoritmos compactos y muy claros, aunque al costo de usar algo de memoria extra en el stack segment y por consiguiente también un algo de tiempo extra por la gestión del stack. Algunos problemas pueden plantearse en forma directa procesando recursivamente diversos subproblemas menores. Tal es el caso de la sucesión de Fibonacci, en la cual cada término se calcula como la suma de los dos inmediatamente anteriores [esto se expresa con la siguiente relación de recurrencia:  $F(n) = F(n-1) + F(n-2)$  con  $F(0) = 0$  y  $F(1) = 1$ ]. La función sencilla que mostramos a continuación que calcula el término  $n$ -ésimo de la sucesión en forma recursiva:

```
def fibo(n):
    if n <= 1:
        return 1
    return fibo(n-1) + fibo(n-2)
```

Sin embargo, un inconveniente adicional es que la *aplicación directa de dos o más invocaciones recursivas* en el planteo de un algoritmo podría hacer que un mismo subproblema se resuelva más de una vez, incrementando el tiempo total de ejecución. Por ejemplo, para calcular  $fibo(4)$  el árbol de llamadas recursivas es el siguiente:



y puede verse que en este caso, se calcula 2 veces el valor de  $fibo(2)$ , 3 veces el valor de  $fibo(1)$  y 2 veces el valor de  $fibo(0)$ ... con lo que la cantidad de llamadas a funciones para hacer más de una vez el mismo trabajo es de 7 ( $= 2 + 3 + 2$ ).

Suponga que se quiere calcular el valor del sexto término de la sucesión. Analice el árbol de llamadas recursivas que se genera al hacer la invocación  $t = fibo(6)$  ¿Cuántas veces en total la función  $fibo()$  se invoca para hacer más de una vez el mismo trabajo, SIN incluir en ese conteo a las invocaciones para obtener  $fibo(0)$  y  $fibo(1)$ ?

Seleccione una:

- a. 10 ✓ ¡Correcto! Efectivamente:  $fibo(4)$  se invoca 2 veces,  $fibo(3)$  se invoca 3 veces y  $fibo(2)$  se invoca 5 veces...
- b. 11
- c. 0
- d. 12

¡Correcto!

**Pregunta 11**

Correcta

Puntúa 1 sobre 1

¿En cuáles de las siguientes situaciones el uso de *recursividad* está efectivamente recomendado? (Más de una respuesta puede ser válida). Marque todas las que considere correctas).

Seleccione una o más de una:



a.

Recorrido y procesamiento de estructuras de datos no lineales como árboles y grafos. ✓

¡Correcto!



b.

Siempre que se pueda escribir una definición recursiva del problema.



c.

Nunca.



d.

Generación y procesamiento de imágenes y gráficos fractales (figuras compuestas por versiones más simples de la misma figura original). ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Recorrido y procesamiento de estructuras de datos no lineales como árboles y grafos.,

Generación y procesamiento de imágenes y gráficos fractales (figuras compuestas por versiones más simples de la misma figura original).

**Comenzado el** martes, 6 de noviembre de 2018, 15:07

**Estado** Finalizado

**Finalizado en** sábado, 10 de noviembre de 2018, 00:08

**Tiempo empleado** 3 días 9 horas

**Puntos** 20/21

**Calificación** 10 de 10 (95%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿En qué universidad estudió, se doctoró y formó parte de su Academia de Ciencias el matemático *Wilhelm Ackerman*?

Seleccione una:

- a.  
Göttingen ✓  
¡Correcto!
- b.  
Oxford
- c.  
Stanford
- d.  
Padova

¡Correcto!

La respuesta correcta es:

Göttingen

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

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

Seleccione una o más de una:



a.

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.



b.

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!



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!



d.

El origen del sistema de coordenadas se encuentra en el punto inferior izquierdo de la pantalla o de la ventana que se esté usando.

¡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.,

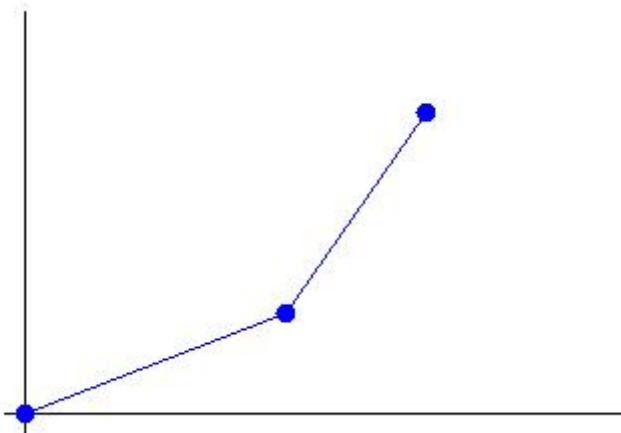
Las filas de pantalla o de ventana con número de orden más bajo, se encuentran más cerca del borde superior que las filas con número de orden más alto.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python el esquema de la gráfica de una función, como la que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función **function(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def function(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    function(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **function(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

- a.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='black')
    canvas.create_line((100, 410, 100, 200), fill='black')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='blue')
    canvas.create_line((230, 350, 300, 250), fill='blue')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='blue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline='blue', fill='blue')
```



¡Correcto!

b.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='orange')
    canvas.create_line((100, 410, 100, 200), fill='orange')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='black')
    canvas.create_line((230, 350, 300, 250), fill='black')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='blue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline='blue', fill='blue')
```

c.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='black')
    canvas.create_line((100, 410, 100, 200), fill='black')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='black', width=3, dash=(5, 4))
    canvas.create_line((230, 350, 300, 250), fill='black', width=3, dash=(5, 4))

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='blue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline='blue', fill='blue')
```

● d.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='black')
    canvas.create_line((100, 410, 100, 200), fill='black')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='black')
    canvas.create_line((230, 350, 300, 250), fill='black')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='blue', fill='red')
    canvas.create_oval((225, 345, 235, 355), outline='blue', fill='red')
    canvas.create_oval((295, 245, 305, 255), outline='blue', fill='red')
```

La respuesta correcta es:

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='black')
    canvas.create_line((100, 410, 100, 200), fill='black')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='blue')
    canvas.create_line((230, 350, 300, 250), fill='blue')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='blue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline='blue', fill='blue')
```

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar Python *el dibujo de un botón para una aplicación que simule una interfaz visual de usuario*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función **button(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def button(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, width=maxw, height=maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    button(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **button(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='red')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 + 1, text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='dark gray')
```

b.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 + 1, text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='dark gray')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='dark gray')
```

c.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 + 1, text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='dark gray')
```



¡Correcto!

d.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 + 1, text='Exit', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='dark gray')
```

¡Correcto!

La respuesta correcta es:

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

        # el fondo y el texto del botón...
        canvas.create_rectangle((x, y, x+ancho, y+alto), fill
='light gray')
        canvas.create_text(x + ancho//2 - 1, y + alto//2 + 1,
text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fi
ll='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fi
ll='white')

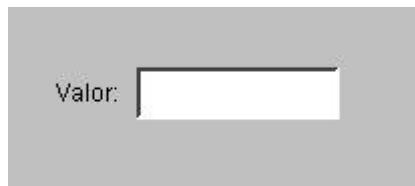
        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+a
lto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-
f, y+f), fill='dark gray')
```

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar Python *el dibujo de un campo de edición de textos para una aplicación que simule una interfaz visual de usuario*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función **edit(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def edit(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, width=maxw, height=maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    edit(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **edit(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:', fill='red')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='white')
```

b.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:', fill='black')

    for f in range(6):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='white')
```

c.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:', fill='black')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fill='white')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f, y+f), fill='dark gray')
```

d.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:', fill='black')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
                           fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
                           fill='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
                           y+alto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-f,
                           y+f), fill='white')
```



¡Correcto!

¡Correcto!

La respuesta correcta es:

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

        # el fondo y el texto del campo de edición...
        canvas.create_rectangle((x, y, x+ancho, y+alto), fill
='white')
        canvas.create_text(x-25, y+alto//2, text='Valor: ', f
ill='black')

        for f in range(2):
            # los bordes oscuros...
            canvas.create_line((x+f, y+f, x+ancho-f, y+f), fi
ll='dark gray')
            canvas.create_line((x+f, y+f, x+f, y+alto-f), fi
ll='dark gray')

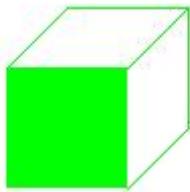
            # los bordes claros...
            canvas.create_line((x+f, y+alto-f, x+ancho-f, y+a
lto-f), fill='white')
            canvas.create_line((x+ancho-f, y+alto-f, x+ancho-
f, y+f), fill='white')
```

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar Python *el dibujo de un cubo* como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función ***cube(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def cube(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    cube(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función ***cube(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='navy', fill='navy')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

b.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='lime', fill='lime', stipple='gray12')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

c.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='lime', fill='lime')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

d.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline='lime', fill='lime')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```



¡Correcto!

¡Correcto!

La respuesta correcta es:

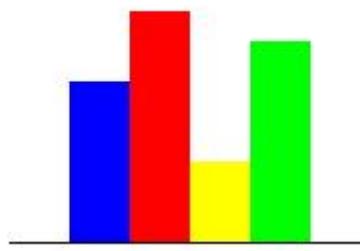
```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline='lime', fill='lime')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python un esquema estadístico tipo *diagrama de barras*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la **bars(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def bars(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    bars(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **bars(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blue', fill='blue')
    canvas.create_rectangle((130, 85, 160, 200), outline='red', fill='red')
    canvas.create_rectangle((161, 160, 190, 200), outline='yellow', fill='yellow')
    canvas.create_rectangle((190, 100, 220, 200), outline='lime', fill='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```



¡Correcto!

b.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blue', fill='blue')
    canvas.create_rectangle((130, 120, 160, 200), outline='red', fill='red')
    canvas.create_rectangle((161, 120, 190, 200), outline='yellow', fill='yellow')
    canvas.create_rectangle((190, 120, 220, 200), outline='lime', fill='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

c.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='red', fill='red')
    canvas.create_rectangle((130, 85, 160, 200), outline='yellow', fill='yellow')
    canvas.create_rectangle((161, 160, 190, 200), outline='lime', fill='lime')
    canvas.create_rectangle((190, 100, 220, 200), outline='blue', fill='blue')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

d.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blue')
    canvas.create_rectangle((130, 85, 160, 200), outline='red')
    canvas.create_rectangle((161, 160, 190, 200), outline='yellow')
    canvas.create_rectangle((190, 100, 220, 200), outline='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

¡Correcto!

La respuesta correcta es:

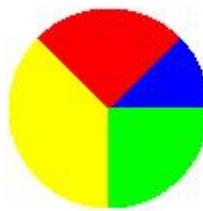
```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blue',
                           fill='blue')
    canvas.create_rectangle((130, 85, 160, 200), outline='red',
                           fill='red')
    canvas.create_rectangle((161, 160, 190, 200), outline='yellow',
                           fill='yellow')
    canvas.create_rectangle((190, 100, 220, 200), outline='lime',
                           fill='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python un esquema estadístico tipo *diagrama de segmentos circulares*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la **pie(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def pie(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    pie(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **pie(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent
=90, outline='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, extent
=135, outline='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, extent
=90, outline='lime', style=PIESLICE)
```

b.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', fill='blue', style=ARC)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent
=90, outline='red', fill='red', style=ARC)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, extent
=135, outline='yellow', fill='yellow', style=ARC)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, extent
=90, outline='lime', fill='lime', style=ARC)
```

c.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 300, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', fill='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent
=90, outline='red', fill='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, extent
=135, outline='yellow', fill='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, extent
=90, outline='lime', fill='lime', style=PIESLICE)
```

d.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', fill='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=90, outline='red', fill='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, exte
nt=135, outline='yellow', fill='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, exte
nt=90, outline='lime', fill='lime', style=PIESLICE)
```



¡Correcto!

¡Correcto!

La respuesta correcta es:

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent=4
5, outline='blue', fill='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=9
0, outline='red', fill='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, extent=
135, outline='yellow', fill='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, extent=
90, outline='lime', fill='lime', style=PIESLICE)
```

**Pregunta 9**

Incorrecta

Puntúa 0 sobre 1

Suponga que se desea generar en Python el dibujo del *esquema básico de una cara*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la **face(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def face(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    face(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **face(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=
180, outline='blue', style=ARC)
```



Incorrecto...

b.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=
180, outline='blue', style=ARC)
```

c.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=0, extent=18
0, outline='blue', style=ARC)
```

d.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red', fill='pink')
    canvas.create_oval((110, 115, 125, 125), outline='blue', fill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', fill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=180, outline='blue', style=ARC)
```

Incorrecto... revise la Ficha 27, página 553 y siguientes...

La respuesta correcta es:

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', fill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', fill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=180, outline='blue', style=ARC)
```

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python el *dibujo del popular Pacman*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la **pacman(canvas)** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *

def pacman(canvas):
    pass

def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
    # de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    pacman(canvas)

    # lanzar el ciclo principal de control de eventos de
    # la ventana...
    root.mainloop()

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

¿Cuál de las siguientes versiones de la función **pacman(canvas)** permitiría dibujar la gráfica sugerida?

Seleccione una:

- a.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=359, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outline='black', fill='black')
```

- b.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=315, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//3), outline='red', fill='red')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outline='black', fill='black')
```

- c.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=315, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outline='black', fill='black')
```



¡Correcto!

- d.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=315, outline='red', fill='red', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outline='black', fill='black')
```

¡Correcto!

La respuesta correcta es:

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=315, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outline='black', fill='black')
```

**Pregunta 11**

Correcta

Puntúa 1 sobre 1

¿Cuál es la mejora esencial que el algoritmo *Quicksort* realiza sobre el algoritmo *Bubblesort* o *Burbuja*?

Seleccione una:

a.

En las versiones analizadas en clases, *Quicksort* sólo usa un ciclo para recorrer el arreglo, mientras que *Bubblesort* usa dos.

b.

*Quicksort* primero determina qué tan desordenado está el arreglo, mientras que *Bubblesort* procede directamente a ordenarlo

c.

*Quicksort* acelera el cambio de posición tanto de los elementos menores como de los mayores, mientras que *Bubblesort* solo acelera a los mayores (o a los menores, dependiendo de la forma de implementación). 

¡Correcto! Justamente por eso *Quicksort* recorre el arreglo desde ambos extremos, para poder hacer intercambios a mayor distancia.

d.

*Quicksort* no implementa ninguna mejora sustancial sobre *Bubblesort*.

¡Correcto!

La respuesta correcta es:

*Quicksort* acelera el cambio de posición tanto de los elementos menores como de los mayores, mientras que *Bubblesort* solo acelera a los mayores (o a los menores, dependiendo de la forma de implementación).

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes esquemas simples de pseudocódigo representa a un algoritmo planteado mediante estrategia Divide y Vencerás, pero con tiempo de ejecución  $O(n * \log(n))$ ?

Seleccione una:

a.

```
proceso(partición) :  
    adicional() [==> O(1)]  
    proceso(partición/2)  
    proceso(partición/2)
```

b.

```
proceso(partición) :  
    adicional() [==> O(n2)]  
    proceso(partición/2)  
    proceso(partición/2)
```

c.

```
proceso(partición) :  
    adicional() [==> O(n)]  
    proceso(partición/2)  
    proceso(partición/2) ✓
```

¡Correcto!

d.

```
proceso(partición) :  
    adicional() [==> O(n3)]  
    proceso(partición/2)  
    proceso(partición/2)
```

¡Correcto!

La respuesta correcta es:

```
proceso(partición) :  
    adicional() [==> O(n)]  
    proceso(partición/2)  
    proceso(partición/2)
```

**Pregunta 13**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones haría que el algoritmo *Quicksort* degenera en su peor caso en cuanto al tiempo de ejecución, de orden  $n^2$ ?

Seleccione una:

a.

Que el arreglo de entrada tenga sus elementos dispuestos de tal forma que cada vez que se seleccione el pivot en cada partición, resulte que ese pivot sea siempre el menor o el mayor de la partición que se está procesando. ✓

¡Correcto!

b.

Que el arreglo esté ya ordenado, pero al revés.

c.

El algoritmo Quicksort no tiene un peor caso  $O(n^2)$ . Su tiempo de ejecución siempre es  $O(n \log(n))$ .

d.

Que el arreglo esté ya ordenado, en la misma secuencia en que se lo quiere ordenar.

¡Correcto!

La respuesta correcta es:

Que el arreglo de entrada tenga sus elementos dispuestos de tal forma que cada vez que se seleccione el pivot en cada partición, resulte que ese pivot sea siempre el menor o el mayor de la partición que se está procesando.

**Pregunta 14**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes estrategias de obtención del pivot es la más recomendable para evitar que el algoritmo Quicksort se degrade en su peor caso  $O(n^2)$  en cuanto a su tiempo de ejecución?

Seleccione una:

a.

En cada partición, usar como pivot al valor de la última casilla.

b.

En cada partición, usar como pivot al valor de la casilla central.

c.

En cada partición, obtener el pivot por la mediana de tres (ya sea la mediana entre el primero, el último y el central; o bien la mediana entre tres elementos aleatorios la partición). 

¡Correcto!

d.

En cada partición, usar como pivot al valor de la primera casilla.

¡Correcto!

La respuesta correcta es:

En cada partición, obtener el pivot por la mediana de tres (ya sea la mediana entre el primero, el último y el central; o bien la mediana entre tres elementos aleatorios la partición).

**Pregunta 15**

Correcta

Puntúa 1 sobre 1

¿Por qué es considerada una *mala idea* tomar como pivot al *primer elemento* (o al *último*) de cada partición al implementar el algoritmo *Quicksort*?

Seleccione una:

- a.

Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese ya ordenado o casi ordenado. ✓

¡Correcto!

- b.

Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el tamaño  $n$  del arreglo fuese muy grande.

- c.

No es cierto que sea una mala idea. Ambas alternativas son tan buenas como cualquier otra.

- d.

Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese completamente desordenado.

¡Correcto!

La respuesta correcta es:

Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese ya ordenado o casi ordenado.

**Pregunta 16**

Correcta

Puntúa 2 sobre 2

Considere la función presentada en clases para calcular mediana entre el primero, el central y el último elemento de una partición del arreglo  $v$  delimitada por los elementos en las posiciones  $izq$  y  $der$ :

```
def get_pivot_m3(v, izq, der):
    # calculo del pivot: mediana de tres...
    central = int((izq + der) / 2)

    if v[der] < v[izq]:
        v[der], v[izq] = v[izq], v[der]

    if v[central] < v[izq]:
        v[central], v[izq] = v[izq], v[central]

    if v[central] > v[der]:
        v[central], v[der] = v[der], v[central]

    return v[central]
```

El tamaño de la partición analizada es entonces  $n = der - izq + 1$  elementos. ¿Cuál es el tiempo de ejecución de esta función, expresado en notación  $O$  y de acuerdo a ese valor de  $n$ ?

Seleccione una:

a.  
 $O(\log(n))$

b.  
 $O(1)$  ✓

¡Correcto! Efectivamente, el cálculo no depende del tamaño de la partición, por lo que resulta de tiempo constante.

c.  
 $O(n)$

d.  
 $O(n^2)$

¡Correcto!

La respuesta correcta es:

$O(1)$

**Pregunta 17**

Correcta

Puntúa 2 sobre 2

¿Cuál es la cantidad de niveles del *árbol de invocaciones recursivas* que se genera al ejecutar el *Quicksort* para ordenar un arreglo de  $n$  elementos, en el **caso promedio**? (Es decir: ¿Cuál es la *altura* de ese árbol en el **caso promedio**?)

Seleccione una:

a.

$$\text{Altura} = n^2$$

b.

$$\text{Altura} = n$$

c.

$$\text{Altura} = n * \log(n)$$

d.

$$\text{Altura} = \log(n) \quad \checkmark$$

¡Correcto!

¡Correcto!

La respuesta correcta es:

$$\text{Altura} = \log(n)$$

**Pregunta 18**

Correcta

Puntúa 2 sobre 2

¿Cuál es la cantidad de niveles del árbol de invocaciones recursivas que se genera al ejecutar el *Quicksort* para ordenar un arreglo de  $n$  elementos, en el **peor caso**? (Es decir: ¿Cuál es la *altura* de ese árbol en ese peor caso?)

Seleccione una:

a.

$$\text{Altura} = n * \log(n)$$

b.

$$\text{Altura} = \log(n)$$

c.

$$\text{Altura} = n \checkmark$$

¡Correcto!

d.

$$\text{Altura} = n^2$$

¡Correcto!

La respuesta correcta es:

$$\text{Altura} = n$$

---

**Comenzado el** martes, 6 de noviembre de 2018, 15:06

**Estado** Finalizado

**Finalizado en** miércoles, 7 de noviembre de 2018, 22:31

**Tiempo empleado** 1 día 7 horas

**Puntos** 14/14

**Calificación** 10 de 10 (100%)

---

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son correctas en relación a conceptos elementales del análisis de algoritmos? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

a.

El análisis del *caso promedio* es aquel en el cual se estudia el comportamiento de un algoritmo cuando debe procesar una configuración de datos que llegan en forma aleatoria. ✓

¡Correcto!

b.

El análisis del *peor caso* es aquel en el cual se estudia el comportamiento de un algoritmo cuando debe procesar la configuración más desfavorable posible de los datos que recibe. ✓

¡Correcto!

c.

La notación Big O se usa para expresar el rendimiento de un algoritmo en términos de una función que imponga una cota inferior para ese algoritmo en cuanto al factor medido (tiempo o espacio de memoria).

d.

Los dos factores de eficiencia más comúnmente utilizados en el análisis de algoritmos son el tiempo de ejecución de un algoritmo y el espacio de memoria que un algoritmo emplea. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Los dos factores de eficiencia más comúnmente utilizados en el análisis de algoritmos son el tiempo de ejecución de un algoritmo y el espacio de memoria que un algoritmo emplea.,

El análisis del *peor caso* es aquel en el cual se estudia el comportamiento de un algoritmo cuando debe procesar la configuración más desfavorable posible de los datos que recibe.,

El análisis del *caso promedio* es aquel en el cual se estudia el comportamiento de un algoritmo cuando debe procesar una configuración de datos que llegan en forma aleatoria.

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

Dado un arreglo de  $n$  componentes... ¿qué significa decir que en el peor caso la cantidad de comparaciones que realiza el algoritmo de *búsqueda secuencial* es  $O(n)$  (o sea: del orden de  $n$ )?

Seleccione una:

- a.

Significa que en el peor caso el algoritmo no hará ninguna comparación.

- b.

Significa que en el peor caso el algoritmo siempre hará menos de  $n$  comparaciones.

- c.

Significa que en el peor caso el algoritmo hará siempre más de  $n$  comparaciones.

- d.

Significa que en el peor caso el algoritmo hará  $n$  comparaciones. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

Significa que en el peor caso el algoritmo hará  $n$  comparaciones.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

Para cada uno de los algoritmos básicos y/o técnicas de procesamiento generales que se indican en la columna de la izquierda, seleccione la expresión en notación *Big O* que mejor expresa el tiempo de ejecución de ese algoritmo en el peor caso:

Búsqueda secuencial en un arreglo (ordenado o desordenado).

O(n)

Ordenamiento rápido (Quicksort) (Considere aquí el tiempo para el caso promedio).

O( $n^*\log(n)$ )

Multiplicación de matrices cuadradas de tamaño  $n*n$ .

O( $n^3$ ) (n al cubo)

Ordenamiento por selección directa.

O( $n^2$ ) (n al cuadrado)

Acceso directo a un casillero de un vector.

O(1)

Búsqueda binaria en un arreglo ya ordenado.

O( $\log(n)$ )

¡Correcto!

La respuesta correcta es: Búsqueda secuencial en un arreglo (ordenado o desordenado). → O(n), Ordenamiento rápido (Quicksort) (Considere aquí el tiempo para el caso promedio). → O( $n^*\log(n)$ ), Multiplicación de matrices cuadradas de tamaño  $n*n$ . → O( $n^3$ ) (n al cubo), Ordenamiento por selección directa. → O( $n^2$ ) (n al cuadrado), Acceso directo a un casillero de un vector. → O(1), Búsqueda binaria en un arreglo ya ordenado. → O( $\log(n)$ )

**Pregunta 4**

Correcta

Puntúa 1 sobre 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.

La estrategia de *Backtracking* es de base recursiva y permite implementar soluciones de prueba y error explorando las distintas soluciones y volviendo 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!

 b.

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. ✓

¡Correcto!

 c.

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!

 d.

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.

¡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 volviendo 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 5**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **falsa** en relación a las *Estrategias de Resolución de Problemas* (o *Estrategias de Planteo de Algoritmos*)?

Seleccione una:

a.

Son técnicas y recomendaciones para el planteo de problemas que garantizan que se encontrará una solución, empleando la estrategia correcta para cada problema que se enfrente. ✓

¡Correcto! Justamente, esta afirmación es falsa... ninguna técnica garantiza el éxito en el campo de la resolución de problemas...

b.

Se trata de un conjunto de técnicas diversas que podrían ayudar a encontrar la solución a un problema, pero sin garantía de éxito, y aún si se llega a una solución, tampoco se garantiza que esa solución sea eficiente.

c.

Se trata de un conjunto de técnicas diversas que podrían ayudar a encontrar la solución a un problema, pero sin garantía de éxito.

d.

Un mismo problema podría ser resultado en base a dos o más estrategias de resolución diferentes, dando lugar a distintos algoritmos para ese mismo problema.

¡Correcto!

La respuesta correcta es:

Son técnicas y recomendaciones para el planteo de problemas que garantizan que se encontrará una solución, empleando la estrategia correcta para cada problema que se enfrente.

**Pregunta 6**

Correcta

Puntúa 1 sobre 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):

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

**¡Correcto!**

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

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

¿Cuáles de los siguientes son conocidos algoritmos basados en la estrategia *Divide y Vencerás*? (Más de una respuesta puede ser correcta, por lo que marque todas las que considere válidas)

Seleccione una o más de una:

a.

Algoritmo *Quicksort* para ordenamiento de arreglos. ✓

¡Correcto!

b.

Algoritmo *Shellsort* para ordenamiento de arreglos.

c.

Algoritmo *Heapsort* para ordenamiento de arreglos.

d.

Algoritmo *Mergesort* para ordenamiento de arreglos. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Algoritmo *Quicksort* para ordenamiento de arreglos.,

Algoritmo *Mergesort* para ordenamiento de arreglos.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes resume en forma correcta la idea general de la estrategia *Divide y Vencerás* para resolución de problemas?

Seleccione una:

a.

Se usa una tabla para almacenar los resultados de los subproblemas que se hayan calculado, y luego cuando algún subproblema vuelve a aparecer se toma su valor desde la tabla, para evitar pérdida de tiempo.

b.

El conjunto de  $n$  datos se divide en subconjuntos de cualquier tamaño, sin importar si los tamaños de cada subconjunto coinciden entre sí. Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final.

c.

Se aplica una regla simple que parece ser beneficiosa, sin volver atrás ni medir las consecuencias de aplicar esa regla, con la esperanza de lograr el resultado óptimo al final.

d.

El conjunto de  $n$  datos se divide en subconjuntos de aproximadamente el mismo tamaño ( $n/2$ ,  $n/3$ ,  $n/4$ , etc.). Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final. ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

El conjunto de  $n$  datos se divide en subconjuntos de aproximadamente el mismo tamaño ( $n/2$ ,  $n/3$ ,  $n/4$ , etc.). Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

Considere el problema del Cambio de Monedas analizado en clases, y la solución mediante un Algoritmo Ávido también presentada en clases. ¿Cuáles de las siguientes afirmaciones **son ciertas** en relación al problema y al algoritmo citado? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

- a.

Si el Problema de Cambio de Monedas no puede resolverse en forma óptima para un conjunto dado de monedas que incluya a la de 1 centavo, mediante el Algoritmo Ávido propuesto, entonces el problema no tiene solución.

- b.

Sea cual sea el algoritmo que se emplee, es exigible que exista la moneda de 1 centavo, pues de otro modo no habrá solución posible para muchos valores de cambio. ✓

¡Correcto!

- c.

El Algoritmo Ávido sugerido para el Problema del Cambio de Monedas falla si el valor  $x$  a cambiar tiene una moneda igual a  $x$  en el conjunto de valores nominales: en ese caso, el algoritmo provoca un error de runtime y se interrumpe.

- d.

El Algoritmo Ávido sugerido para el problema del Cambio de Monedas funciona correctamente para cualquier conjunto de valores nominales de monedas, siempre y cuando ese conjunto incluya a la moneda de 1 centavo.

¡Correcto!

La respuesta correcta es:

Sea cual sea el algoritmo que se emplee, es exigible que exista la moneda de 1 centavo, pues de otro modo no habrá solución posible para muchos valores de cambio.

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

Considere el problema del *Cambio de Monedas* analizado en clases, y la solución mediante *Programación Dinámica* también presentada en clases ¿Cuáles de las siguientes afirmaciones **son ciertas** en relación al problema y al algoritmo citado? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:



a.

En el algoritmo basado en Programación Dinámica, el resultado final a retornar es el que haya quedado almacenado en la casilla  $x$  del arreglo *prev* que contiene los resultados intermedios (o sea, en *prev[x]*). ✓

¡Correcto!



b.

En el algoritmo basado en Programación Dinámica, el resultado final a retornar es igual a la suma o acumulación de todos los valores almacenados en el arreglo *prev* donde se almacenaron los resultados intermedios.



c.

En el algoritmo basado en Programación Dinámica no es importante si el arreglo *coins* está ordenado o desordenado: funcionará correctamente de todas formas ✓

¡Correcto!



d.

En el algoritmo basado en Programación Dinámica los valores de las monedas que sean mayores a  $x$ , son dejados de lado y la recurrencia de cálculo no se aplica sobre ellos. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

En el algoritmo basado en Programación Dinámica no es importante si el arreglo *coins* está ordenado o desordenado: funcionará correctamente de todas formas,

En el algoritmo basado en Programación Dinámica los valores de las monedas que sean mayores a  $x$ , son dejados de lado y la recurrencia de cálculo no se aplica sobre ellos.,

En el algoritmo basado en Programación Dinámica, el resultado final a retornar es el que haya quedado almacenado en la casilla  $x$  del arreglo *prev* que contiene los resultados intermedios (o sea, en *prev[x]*).

**Pregunta 11**

Correcta

Puntúa 1 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. ¿Cuáles de las siguientes afirmaciones son **ciertas** en relación a las **diagonales del tablero** en el cual deben colocarse la reinas, suponiendo que el tablero es el normal del ajedrez, de  $8 \times 8$ ? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

 a.

En general hay dos tipos de diagonales: las normales (orientadas en la misma forma que la diagonal principal) y las inversas (orientadas en la misma forma que la contra-diagonal o diagonal inversa).

¡Correcto!

 b.

En cada una de las diagonales que se orientan como la contra-diagonal o diagonal inversa, es constante la suma entre el número de columna y el número de fila de cada uno de sus elementos.

¡Correcto!

 c.

La cantidad **total** de diagonales que contiene el tablero (sumando todas las diagonales de todos los tipos posibles) es 30.

¡Correcto! Efectivamente... son 15 normales y 15 inversas...

 d.

En cada una de las diagonales que se orientan como la principal, es constante la resta entre el número de columna y el número de fila de cada uno de sus elementos.

¡Correcto!

¡Correcto!

Las respuestas correctas son:

En general hay dos tipos de diagonales: las normales (orientadas en la misma forma que la diagonal principal) y las inversas (orientadas en la misma forma que la contra-diagonal o diagonal inversa).,

La cantidad **total** de diagonales que contiene el tablero (sumando todas las diagonales de todos los tipos posibles) es 30.,

En cada una de las diagonales que se orientan como la principal, es constante la resta entre el número de columna y el número de fila de cada uno de sus elementos.,

En cada una de las diagonales que se orientan como la contra-diagonal o diagonal inversa, es constante la suma entre el número de columna y el número de fila de cada uno de sus elementos.

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. ¿Cuáles de las siguientes afirmaciones son **ciertas** en relación a las **diagonales normales del tablero** en el cual deben colocarse la reinas, suponiendo que el tablero es el normal del ajedrez, de  $8 * 8$ ? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:



a.

Las diagonales normales pueden representarse con un arreglo *qnd* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor (*col - fil*), se haga coincidir el casillero *qnd[(col - fil) + 7]* (evitando de esta forma los índices negativos). 

¡Correcto!



b.

Las diagonales normales pueden representarse con un arreglo *qnd* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor (*col + fil*), se haga coincidir el casillero *qnd[(col + fil)]*.



c.

El valor de la suma entre el número de columna y el número de fila de cada componente de una diagonal normal, es un número constante para cada diagonal, y los posibles valores están en el intervalo [0..14]



d.

El valor de la resta entre el número de columna y el número de fila de cada componente de una diagonal normal, es un número constante para cada diagonal, y los posibles valores están en el intervalo [-7..7]



¡Correcto!

¡Correcto!

Las respuestas correctas son:

El valor de la resta entre el número de columna y el número de fila de cada componente de una diagonal normal, es un número constante para cada diagonal, y los posibles valores están en el intervalo [-7..7],

Las diagonales normales pueden representarse con un arreglo *qnd* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor (*col - fil*), se haga coincidir el casillero *qnd[(col - fil) + 7]* (evitando de esta forma los índices negativos).

**Pregunta 13**

Correcta

Puntúa 1 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. ¿Cuáles de las siguientes afirmaciones son **ciertas** en relación a las **diagonales inversas del tablero** en el cual deben colocarse la reinas, suponiendo que el tablero es el normal del ajedrez, de 8 \* 8? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

- a.

El valor de la suma entre el número de columna y el número de fila de cada componente de una diagonal inversa, es un número constante para cada diagonal, y los posibles valores están en el intervalo [0..14]



¡Correcto!

- b.

El valor de la resta entre el número de columna y el número de fila de cada componente de una diagonal inversa, es un número constante para cada diagonal, y los posibles valores están en el intervalo [-7..7]

- c.

Las diagonales inversas pueden representarse con un arreglo *qid* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor (*col + fil*), se haga coincidir el casillero *qid*[(*col + fil*)].



¡Correcto!

- d.

Las diagonales inversas pueden representarse con un arreglo *qid* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor (*col - fil*), se haga coincidir el casillero *qid*[(*col - fil*) + 7].

¡Correcto!

Las respuestas correctas son:

El valor de la suma entre el número de columna y el número de fila de cada componente de una diagonal inversa, es un número constante para cada diagonal, y los posibles valores están en el intervalo [0..14],

Las diagonales inversas pueden representarse con un arreglo *qid* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor (*col + fil*), se haga coincidir el casillero *qid*[(*col + fil*)].

**Pregunta 14**

Correcta

Puntúa 1 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. Se ha indicado que se puede usar un arreglo  $rc$  de componentes, en el cual el casillero  $rc[col] = fil$  indica que la reina de la columna  $col$  está ubicada en la fila  $fil$ . ¿Cuáles de las siguientes configuraciones para el arreglo  $rc$  representan **soluciones incorrectas** para el problema de las *Ocho Reinas*? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

 a. $rc = [3, 5, 7, 0, 5, 1, 2, 4]$  ✓

¡Correcto! Efectivamente, si  $rc$  contiene los valores mostrados, hay al menos un problema: las reinas de las columnas 1 y 4 están ubicadas en la misma fila: la 5.

 b. $rc = [4, 7, 3, 0, 2, 5, 1, 6]$  c. $rc = [5, 3, 6, 0, 7, 1, 4, 2]$  d. $rc = [2, 0, 7, 4, 5, 1, 6, 3]$  ✓

¡Correcto! Efectivamente, si  $rc$  contiene los valores mostrados, **hay al menos** un problema: las reinas de las columnas 3 y 4 están ubicadas en la misma diagonal normal: la diagonal -1.

¡Correcto!

Las respuestas correctas son:

 $rc = [3, 5, 7, 0, 5, 1, 2, 4],$  $rc = [2, 0, 7, 4, 5, 1, 6, 3]$

Comenzado el	martes, 6 de noviembre de 2018, 15:06
Estado	Finalizado
Finalizado en	viernes, 9 de noviembre de 2018, 21:26
Tiempo empleado	3 días 6 horas
Puntos	12/12
Calificación	10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 1 sobre 1

¿Qué se entiende (en el contexto de la Teoría de la Complejidad Computacional) por *Problemas de Decisión*?

Seleccione una:

a.

Son problemas en los que se pide obtener la solución específica para el problema planteado, de forma que la solución puede ser numérica.

b.

Son problemas en los que se pide obtener una solución aproximada (aunque no necesariamente correcta o exacta)

c.

Son problemas en los que sólo se admite como resultado un valor lógico (o valor de verdad: verdadero o falso).

¡Correcto!

d.

Son problemas en los que se pide obtener la mejor solución de entre varias posibles (y no necesariamente *cualquier* solución).

¡Correcto!

La respuesta correcta es:

Son problemas en los que sólo se admite como resultado un valor lógico (o valor de verdad: verdadero o falso).

**Pregunta 2**

Correcta

Puntúa 1 sobre 1

¿Qué se entiende (en el contexto de la Teoría de la Complejidad Computacional) por *Problemas Intratables*?

Seleccione una:

a.

Son problemas para los que no se conoce solución alguna.

b.

Son problemas para los que se conocen soluciones cuyo tiempo de ejecución es  $O(2^n)$  (o alguna otra función o relación superpolinomial), pero para los que *también* se conocen soluciones de tiempo polinomial  $O(n^k)$ , siendo  $n$  el tamaño del problema.

c.

Son problemas para los que sólo se conocen soluciones cuyo tiempo de ejecución es  $O(2^n)$  (o alguna otra función o relación superpolinomial) siendo  $n$  el tamaño del problema.

¡Correcto!

d.

Son problemas para los que sólo se conocen soluciones cuyo tiempo de ejecución es  $O(n^k)$  (o sea, tiempo de "ejecución polinomial") siendo  $n$  el tamaño del problema.

¡Correcto!

La respuesta correcta es:

Son problemas para los que sólo se conocen soluciones cuyo tiempo de ejecución es  $O(2^n)$  (o alguna otra función o relación superpolinomial) siendo  $n$  el tamaño del problema.

**Pregunta 3**

Correcta

Puntúa 1 sobre 1

Suponga que para un problema  $p$  dado, cuyo volumen de entrada es  $n$ , se conocen tres algoritmos diferentes  $a_1$ ,  $a_2$  y  $a_3$  para resolverlo. Suponga que los tiempos de ejecución de estos tres algoritmos son  $t(a_1) = O(2^n)$ ,  $t(a_2) = O(n^4)$  y  $t(a_3) = O(n^{2.5}\log(n))$ . ¿Cuáles de las siguientes afirmaciones son ciertas respecto del problema  $p$  desde el punto de vista de la Teoría de la Complejidad Computacional? (Más de una respuesta puede ser cierta... marque todas las que considere correctas...)

Seleccione una o más de una:



- a.  
El problema  $p$  no es intratable, ya que al menos uno de los algoritmos que se conocen para resolverlo ejecuta en tiempo polinomial o subpolinomial. ✓

¡Correcto!



- b.  
No hay forma de decidir si  $p$  es intratable o no, ya que algunos de sus algoritmos ejecutan en tiempo exponencial y otros lo hacen en tiempo polinomial o subpolinomial.



- c.  
El problema  $p$  es intratable, ya que el algoritmo  $a_1$  ejecuta en tiempo exponencial (considerado impráctico por la teoría).



- d.  
Los algoritmos  $a_2$  y  $a_3$  tienen tiempos de ejecución aceptables para la teoría, por lo tanto  $p$  no es un problema intratable. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Los algoritmos  $a_2$  y  $a_3$  tienen tiempos de ejecución aceptables para la teoría, por lo tanto  $p$  no es un problema intratable.,

El problema  $p$  no es intratable, ya que al menos uno de los algoritmos que se conocen para resolverlo ejecuta en tiempo polinomial o subpolinomial.

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

Suponga que se tienen dos problemas  $p_1$  y  $p_2$ . Sea  $p_2 = R(p_1)$  una reducción del problema  $p_1$  al problema  $p_2$ . ¿Cuáles de las siguientes afirmaciones son ciertas? (Más de una respuesta puede ser válida... marque todas las que considere correctas) Si

Seleccione una o más de una:



- a.  
Si se encuentra un algoritmo que resuelva  $p_2$ , entonces el mismo algoritmo permitirá resolver  $p_1$ . ✓

¡Correcto!



- b.  
Si se encuentra un algoritmo que resuelva  $p_1$ , entonces el mismo algoritmo permitirá resolver  $p_2$ .



- c.  
Si la reducción  $R$  permite reducir  $p_1$  a  $p_2$ , entonces también puede reducir  $p_2$  a  $p_1$ .



- d.  
Si la reducción  $R$  permite reducir  $p_1$  a  $p_2$ , entonces también puede reducir  $p_1$  a cualquier otro problema.

¡Correcto!

La respuesta correcta es:

Si se encuentra un algoritmo que resuelva  $p_2$ , entonces el mismo algoritmo permitirá resolver  $p_1$ .

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

Suponga que se tienen dos problemas  $p_1$  y  $p_2$ , ambos con tamaño de entrada  $n$ . Se conoce un algoritmo  $a_1$  para resolver el problema  $p_1$  y otro algoritmo  $a_2$  para resolver el problema  $p_2$ . Sea  $p_2 = R(p_1)$  una reducción del problema  $p_1$  al problema  $p_2$ . ¿En cuáles de las situaciones que siguen, la reducción  $R$  sería aceptable en la práctica? (Más de una respuesta puede ser válida... marque todas las que considere correctas)

Seleccione una o más de una:

 a.

Tiempo de ejecución de  $a_2$ :  $O(n^4)$

Tiempo de ejecución de  $a_1$ : Asuma que no se conoce ningún algoritmo para resolver  $p_1$

Tiempo de ejecución de  $R$ :  $O(n^4)$  ✓

¡Correcto! Efectivamente: si por el momento no se conoce ningún algoritmo  $a_1$  para resolver  $p_1$ , entonces la reducción  $R$  vale la pena: se obtendría un algoritmo  $a_2$  donde antes no había ninguno. Y en este caso, tendría poca importancia si la propia reducción  $R$  ejecuta en un tiempo alto... Un algoritmo  $O(n^4)$  más una reducción  $O(n^4)$  es mejor que ningún algoritmo...

 b.

Tiempo de ejecución de  $a_2$ :  $O(2^n)$

Tiempo de ejecución de  $a_1$ :  $O(2^n)$

Tiempo de ejecución de  $R$ :  $O(n^k)$  para algún  $k$  entero  $\geq 0$ . ✓

¡Correcto! Efectivamente: En este caso la reducción  $R$  es polinómica (ejecuta en tiempo polinomial), y aunque pudiera parecer que no se gana nada en reducir  $p_1$  a  $p_2$  (ya que ambos son de tiempo exponencial), lo real es que tampoco se pierde nada y queda un margen de esperanza: si se llegase a encontrar una buena solución para  $p_2$ , esa misma solución resolvería también  $p_1$  al costo de una reducción de tiempo aceptable.

 c.

Tiempo de ejecución de  $a_2$ :  $O(n^2)$

Tiempo de ejecución de  $a_1$ :  $O(n^3)$

Tiempo de ejecución de  $R$ :  $O(n)$  ✓

¡Correcto! Efectivamente:  $a_2$  es mejor algoritmo que  $a_1$  (en tiempo de ejecución), y el tiempo para reducir  $p_1$  a  $p_2$  es menor que el tiempo de ejecución de  $a_2$ ... ¡La reducción  $R$  vale la pena!

 d.

Tiempo de ejecución de  $a_2$ :  $O(n^2)$

Tiempo de ejecución de  $a_1$ :  $O(\log(n))$

Tiempo de ejecución de  $R$ :  $O(n)$

¡Correcto!

Las respuestas correctas son:

Tiempo de ejecución de  $a_2$ :  $O(n^2)$

Tiempo de ejecución de  $a_1$ :  $O(n^3)$

Tiempo de ejecución de  $R$ :  $O(n)$ ,

Tiempo de ejecución de  $a_2$ :  $O(n^4)$

Tiempo de ejecución de  $a_1$ : Asuma que no se conoce ningún algoritmo para resolver  $p_1$

Tiempo de ejecución de  $R$ :  $O(n^4)$ ,

Tiempo de ejecución de  $a_2$ :  $O(2^n)$

Tiempo de ejecución de  $a_1$ :  $O(2^n)$

Tiempo de ejecución de  $R$ :  $O(n^k)$  para algún  $k$  entero  $\geq 0$ .

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

Consideré el *Problema del Clique Máximo* para un grafo de  $n$  vértices, representado en forma matricial, tal como se explicó en clases y en los materiales de consulta. Hemos indicado que el único algoritmo conocido para resolver el *Problema del Clique* es de tiempo  $O(n^2 * 2^n)$  y por lo tanto es un *problema intratable*. ¿Cuáles de las siguientes son ciertas si se encontrase un nuevo algoritmo para resolver el *Problema del Clique Máximo*, pero tal que el tiempo de ejecución de este nuevo algoritmo sea  $O(n^k)$  (para  $k$  entero y  $k \geq 0$ )? (Más de una respuesta puede ser válida. Marque todas las que considere correctas).

Seleccione una o más de una:

a.

El Problema del Clique dejaría de ser considerado intratable. ✓

¡Correcto!

b.

Cualquier otro problema considerado intratable que pueda ser reducido al Problema del Clique Máximo mediante una reducción polinómica, también dejaría de ser intratable ✓

¡Correcto!

c.

El Problema del Clique seguiría siendo considerado intratable.

d.

La situación planteada no tiene sentido. Si una de las soluciones para el Problema del Clique es de tiempo exponencial, entonces todas las posibles nuevas soluciones serán también de tiempo de ejecución exponencial.

¡Correcto!

Las respuestas correctas son:

El Problema del Clique dejaría de ser considerado intratable.,

Cualquier otro problema considerado intratable que pueda ser reducido al Problema del Clique Máximo mediante una reducción polinómica, también dejaría de ser intratable

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Desde el punto de vista de la Teoría de la Complejidad los problemas pueden ser clasificados de acuerdo al tipo de **Máquina Teórica** para la cual se admitan soluciones para esos problemas. ¿Cuáles de las siguientes afirmaciones sobre estas Máquinas Teóricas son ciertas? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

a.

Las Máquinas Teóricas No Deterministas tienen comportamiento aleatorio: Para un mismo problema estas máquinas podrían obtener resultados diferentes.

b.

Todas las computadoras existentes hasta hoy, están basadas en un modelo de Máquina Teórica conocido como **Máquina de Turing**.



¡Correcto!

c.

En una Máquina Teórica No Determinista es imposible determinar el siguiente estado en el que estará esa máquina sólo sabiendo el estado actual de la misma. ✓

¡Correcto!

d.

En una Máquina Teórica Determinista el siguiente estado en el que estará esa máquina depende por completo del estado actual de la misma. Por lo tanto, el siguiente estado puede determinarse con precisión. ✓

¡Correcto!

¡Correcto!

Las respuestas correctas son:

Todas las computadoras existentes hasta hoy, están basadas en un modelo de Máquina Teórica conocido como **Máquina de Turing**.,

En una Máquina Teórica No Determinista es imposible determinar el siguiente estado en el que estará esa máquina sólo sabiendo el estado actual de la misma.,

En una Máquina Teórica Determinista el siguiente estado en el que estará esa máquina depende por completo del estado actual de la misma. Por lo tanto, el siguiente estado puede determinarse con precisión.

**Pregunta 8**

Correcta

Puntúa 1 sobre 1

Para cada una de las clases de complejidad que aparecen en la columna de la izquierda, seleccione la definición que mejor se le ajuste.

- |            |   |
|------------|---|
| Clase NPC  | Conjunto de todos los problemas NP-Complete.  |
| Clase NEXP | Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo exponencial usando una máquina no determinista. |
| Clase EXP  | Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo exponencial usando una máquina determinista.    |
| Clase NP   | Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo polinomial usando una máquina no determinista.  |
| Clase P    | Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo polinomial usando una máquina determinista.     |

¡Correcto!

La respuesta correcta es: Clase NPC → Conjunto de todos los problemas NP-Complete., Clase NEXP → Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo exponencial usando una máquina no determinista., Clase EXP → Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo exponencial usando una máquina determinista.,

Clase NP → Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo polinomial usando una máquina no determinista.,

Clase P → Conjunto de todos los problemas de decisión que pueden ser resueltos con algoritmos de tiempo polinomial usando una máquina determinista.

**Pregunta 9**

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones **son ciertas** en relación a las clases de complejidad **P** y **NP**? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

- a.  
Todos los problemas intratables pertenecen a NP.
- b.  
Ningún problema de NP puede resolverse en tiempo polinomial con una máquina determinista.
- c.  
Por lo que hasta ahora se sabe, si un problema pertenece a NP, entonces también pertenece a P.
- d.  
Por lo que hasta ahora se sabe, si un problema pertenece a P, entonces también pertenece a NP. ✓

¡Correcto! Efectivamente, la clase P es un subconjunto de la clase NP...

¡Correcto!

La respuesta correcta es:

Por lo que hasta ahora se sabe, si un problema pertenece a P, entonces también pertenece a NP.

Pregunta 10

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones serían ciertas si el problema P vs NP se resolviese por la afirmativa (es decir, si se demostrase que P y NP son efectivamente iguales? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

a.

Todo problema actualmente en NP, tendría solución de tiempo polinomial en una máquina determinista. ✓

¡Correcto!

b.

Todo problema actualmente en P admitiría también soluciones de tiempo exponencial en las mismas máquinas *deterministas* para las que esos problemas ya tenían soluciones de tiempo polinomial.

c.

Las máquinas deterministas serían al menos tan potentes como las no deterministas: Ambas resolverían eficientemente los mismos tipos de problemas ✓

¡Correcto!

d.

Todo problema actualmente considerado intratable tendría solución de tiempo de ejecución polinomial en una máquina determinista.

¡Correcto!

Las respuestas correctas son:

Todo problema actualmente en NP, tendría solución de tiempo polinomial en una máquina determinista.,

Las máquinas deterministas serían al menos tan potentes como las no deterministas: Ambas resolverían eficientemente los mismos tipos de problemas

Pregunta 11

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones serían ciertas si el problema P vs NP se resolviese por la negativa (es decir, si se demostrase que P y NP no son realmente iguales? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

a.

Todo problema actualmente en P se convertiría en intratable.

b.

Todo problema actualmente en NP, tendría solución de tiempo polinomial en una máquina determinista.

c.

Se tendría la certeza de que al menos para algunos problemas no existen buenas soluciones, lo que permitiría dejar de buscarlas inútilmente y pasar a concentrar los esfuerzos en diseñar soluciones aproximadas o no óptimas. ✓

¡Correcto!

d.

Todo problema actualmente considerado intratable tendría solución de tiempo de ejecución polinomial en una máquina determinista.

¡Correcto!

La respuesta correcta es:

Se tendría la certeza de que al menos para algunos problemas no existen buenas soluciones, lo que permitiría dejar de buscarlas inútilmente y pasar a concentrar los esfuerzos en diseñar soluciones aproximadas o no óptimas.

Pregunta 12

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son ciertas respecto de los problemas NP-Complete? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

a.

Si p1 es un problema NP-Complete, entonces existen reducciones polinómicas que permiten reducir todo problema de P al problema p1. ✓

¡Correcto! Recuerde que P es subconjunto de NP... y si p1 es NP-Complete entonces (por definición) todo problema de NP puede reducirse a p1... incluidos todos los de P..

b.

Si p1 es un problema NP-Complete, entonces existen reducciones polinómicas que permiten reducir todo problema de NP a al problema p1. ✓



¡Correcto! Justamente esa es la idea de un problema NP-Complete...

c.

Si p1 es un problema NP-Complete, entonces existe al menos una reducción polinómica que permite reducir el problema p1 a algún problema de P.

d.

Si p1 es un problema NP-Complete, entonces existe al menos una reducción polinómica que permite reducir el problema p1 a algún problema de NP.

¡Correcto!

Las respuestas correctas son:

Si p1 es un problema NP-Complete, entonces existen reducciones polinómicas que permiten reducir todo problema de NP a al problema p1.

,

Si p1 es un problema NP-Complete, entonces existen reducciones polinómicas que permiten reducir todo problema de P al problema p1.

Página Principal ► Mis cursos ► AED (2018) ► 18 de marzo - 24 de marzo ► Cuestionario 01 [Temas: Ficha 01]

---

**Comenzado el** martes, 27 de marzo de 2018, 19:52

**Estado** Finalizado

**Finalizado en** lunes, 2 de abril de 2018, 14:36

**Tiempo empleado** 5 días 18 horas

**Puntos** 41/42

**Calificación** 10 de 10 (98%)

---

Cuestionario 01 Temas: Ficha 01  
Para cada una de las personas célebres cuyos nombres aparecen a la izquierda, seleccione el aporte principal que dicha persona ha realizado al mundo de las ciencias informáticas o las ciencias exactas.



Thomas  
Flowers

Diseñador de Colossus, la primera máquina operable considerada como antecedente de las computadoras modernas.



Ada Byron

Primeros conceptos fundamentales de programación (subrutinas, ciclos, etc.)



Alan  
Turing

Director del equipo que desarrolló Bombe, la máquina que permitió descifrar el código Enigma alemán.



Abu  
Abdallah  
Muhammad  
ibn Musā  
al-Khwārizmī  
(Abu  
Yāffar)

Primeras reglas algorítmicas para las operaciones aritméticas elementales en números arábigos.



Charles  
Babbage

Diseño de la Analytical Engine (primer diseño práctico de una computadora en el mundo)



¡Ok!



La respuesta correcta es:

Thomas Flowers → Diseñador de Colossus, la primera máquina operable considerada como antecedente de las computadoras

modernas.,



Ada Byron → Primeros conceptos fundamentales de programación (subrutinas, ciclos, etc.).



Alan Turing → Director del equipo que desarrolló Bombe, la máquina que permitió descifrar el código Enigma alemán..



Abu Abdallah Muhammad ibn Musā al-Khwārizmī (Abu Yāffar) → Primeras reglas algorítmicas para las operaciones aritméticas

elementas en números arábigos..



Charles Babbage → Diseño de la Analytical Engine (primer diseño práctico de una computadora en el mundo)

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

¿Cuál es el problema (si lo hay) si se ejecuta el siguiente script en Python 3?

```
n1 = 10
n2 = 14
n1 = int(input('Ingrese un número entero: '))
n2 = float(input('Ingrese un número en coma flotante: '))
print('n1: ', n1)
print('n2: ', n2)
```

Seleccione una:

a.

En Python 3 no hay ninguna función llamada float() para convertir cadenas a números flotantes.

b.

No hay ningún problema. ✓

¡Ok! Si tuvo alguna duda por el hecho de que la variable n2 comenzó con un valor int y luego se le asignó un valor float, no olvide que Python es un lenguaje de tipado dinámico, y por lo tanto una variable puede cambiar de tipo durante la ejecución de un programa.

c.

Si una variable ya fue asignada con un valor, no se puede cambiar ese valor por otro cargado por teclado.

d.

La variable n2 se definió como int al asignarle el valor inicial 14, y luego se le asignó un valor float cargado por teclado: no se puede cambiar el tipo de una variable.

¡Correcto!

La respuesta correcta es:

No hay ningún problema.

6/12/2018

Pregunta 3

Correcta

Puntúa 3 sobre 3

Cuestionario 01 | Temas: Ficha 01  
 ¿Cuántos números enteros diferentes pueden representarse en binario si se dispone de un conjunto de 8(ocho) bytes agrupados?

Seleccione una:

- a.  
 $2^{64}$  (= 18446744073709551616 números enteros diferentes) ✓

¡Ok! Cada bit permite 2 combinaciones, y dispone de (8 bytes \* 8 bits cada uno) = 64 bits en total, por lo que la cantidad total de combinaciones (o sea, la cantidad de números diferentes que pueden representarse) es  $2^{64}$ .

- b.  
 $64^2$  (= 4096 números enteros diferentes)
- c.  
 $2^{16}$  (= 65536 números enteros diferentes)
- d.  
 $2^8$  (= 256 números enteros diferentes)

¡Correcto!

La respuesta correcta es:

$2^{64}$  (= 18446744073709551616 números enteros diferentes)

Pregunta 4

Correcta

Puntúa 2 sobre 2

Suponga la siguiente instrucción de carga por teclado en Python 3:

```
x = float(input('Ingrese un numero: '))
```

¿Cuál de las siguientes afirmaciones es **CIERTA**?

Seleccione una:

- a.  
 Si se ingresa por teclado número entero, se producirá un error y la ejecución del script se interrumpirá.
- b.  
 Si se ingresa por teclado un valor que no puede convertirse a un número, la variable **x** quedará valiendo **None**.
- c.  
 Si se ingresa por teclado un número entero, la variable **x** quedará valiendo el valor **None**.
- d.  
 Si se ingresa por teclado un valor que no puede convertirse a un número, se producirá un error y la ejecución del script se interrumpirá. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

Si se ingresa por teclado un valor que no puede convertirse a un número, se producirá un error y la ejecución del script se interrumpirá.

6/12/2018

Pregunta 5

Correcta

Puntúa 2 sobre 2

Cuestionario 01 Temas: Ficha 01  
Analice el siguiente script simple, cuyo objetivo es tomar por teclado los datos básicos de un postulante a un crédito, y mostrar por consola estándar los datos cargados:

```
a = input('Ingrese su nombre: ')
print('El nombre ingresado es: ', a)
a = int(input('Ahora ingrese su edad: '))
print('La edad ingresada es: ', a)
a = float( input('Y ahora ingrese sueldo: ') )
print('El sueldo ingresado es: ', a)
```

¿Producirá algún problema la ejecución de este script?

Seleccione una:

a.

Sí. El script ejecutará sin problemas aparentes, pero mostrará en consola estándar siempre el mismo valor: el valor *None*.

b.

Sí. El script ejecutará sin problemas aparentes, pero mostrará en consola estándar siempre el mismo valor: el nombre del postulante.

c.

No. El script se ejecutará sin problemas y hará lo esperado. ✓

¡Ok!

d.

Sí. El script comenzará a ejecutarse, pero lanzará un error y se interrumpirá cuando intente cargar la edad del postulante en la variable a que ya contenía el nombre.

¡Correcto!

La respuesta correcta es:

No. El script se ejecutará sin problemas y hará lo esperado.

Pregunta 6

Correcta

Puntúa 1 sobre 1

¿Qué significa **definir una variable** en Python?

Seleccione una:

a. Indicar su nombre y asignarle un valor. ✓ ¡Ok!

b. Indicar su nombre.

c. Indicar su tipo, su nombre y su tamaño.

d. Indicar su tipo y su nombre o identificador.

¡Correcto!

La respuesta correcta es: Indicar su nombre y asignarle un valor.

**Pregunta 7**

Correcta

Puntúa 2 sobre 2

¿Hay algún problema con el siguiente script en *Python* 3?

```
y = x * 2  
print('Valor final: ', y)
```

Seleccione una:

- a.  
Lanza un error: la variable *x* no está definida en el momento en que se multiplica por 2. ✓  
¡Ok!
- b.  
Está mal realizada la visualización del resultado: en Python 3 *print* no debe escribirse con paréntesis.
- c.  
La expresión *y = x \* 2* no tiene sentido en Python.
- d.  
No hay ningún problema.

¡Correcto!

La respuesta correcta es:

Lanza un error: la variable *x* no está definida en el momento en que se multiplica por 2.

**Pregunta 8**

Incorrecta

Puntúa 0 sobre 1

¿Qué diferencia principal hay entre una *calculadora manual común* y una *computadora*?

(Tómese su tiempo para pensar y discutir esta pregunta... No encontrará la respuesta directamente en la Ficha 01).

Seleccione una:

- a.  
Ninguna. ✗  
  
Incorrecto... ¿Puede usar una calculadora manual común en reemplazo de una computadora en todas sus aplicaciones?
- b.  
Las computadoras son programables, mientras que las calculadoras no.
- c.  
Las calculadoras manuales comunes no pueden procesar texto ni otros tipos de datos no numéricos. Las computadoras sí.
- d.  
Las calculadoras no pueden componer ni desplegar imágenes, mientras que las computadoras sí.

La lectura de las dos primeras secciones de la Ficha 01 podría darle pistas sobre la respuesta correcta, pero lo mejor sería discutirla con sus compañeros y/o sus profesores.

La respuesta correcta es:

Las computadoras son programables, mientras que las calculadoras no.

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

Suponga que tiene un script Python almacenado en un archivo llamado "script.py". Suponga también que está trabajando bajo sistema operativo Windows, y que el archivo *script.py* está guardado en la carpeta "C:\Programas". Finalmente, suponga que la variable de entorno *PATH* de Windows contiene correctamente la ruta del intérprete de Python. En estas condiciones, ¿cuál de las siguientes órdenes provocará que el script sea ejecutado desde la línea de órdenes de Windows, asumiendo que la carpeta activa es la que indica en cada caso el prompt?

Seleccione una:

- a.  
C:\Programas>python script
- b.  
C:\Programas>python prueba.py
- c.  
C:\Program Files>python script.py
- d.  
C:\Programas>python script.py ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

C:\Programas>python script.py

**Pregunta 10**

Correcta

Puntúa 1 sobre 1

¿Qué se entiende, en general, por error de compilación?

Seleccione una:

- a.  
Es un error en la *sintaxis* del programa, que provoca que el programa no pueda comenzar a ejecutarse (si es compilado) o no pueda seguir ejecutándose (si es interpretado) al llegar a la línea con ese error. ✓  
¡Ok!
- b.  
Es un error en el *hardware de la computadora*, que provoca una falla grave de funcionamiento de todos los programas.
- c.  
Es un error producido por una *operación imposible de ejecutar*, aunque sintácticamente bien escrita (por ejemplo, una división por cero), que provoca que el programa se interrumpe de forma abrupta y anormal una vez que comenzó a ejecutarse
- d.  
Es un error en la *lógica* del programa, que provoca que al ejecutarse el programa arroje resultados incorrectos.

¡Correcto!

La respuesta correcta es:

Es un error en la *sintaxis* del programa, que provoca que el programa no pueda comenzar a ejecutarse (si es compilado) o no pueda seguir ejecutándose (si es interpretado) al llegar a la línea con ese error.

**Pregunta 11**

Correcta

Puntúa 2 sobre 2

**¿Hay algún error en el siguiente script de instrucciones en Python 3?**

```
nombre = input('Nombre: ')
edad = int(input('Edad: '))
print('Datos recibidos - Nombre: ', Nombre, 'Edad: ', Edad)
```

Seleccione una:

- a.  
No hay ningún error.
- b.  
El error es que la función input() de Python 3 no puede usarse para cargar cadenas de caracteres en forma directa.
- c.  
El error es el uso de la función int() en la segunda carga: no existe tal función en Python 3.
- d.  
**El error es que las variables edad y nombre se definieron en minúsculas al hacer la carga, y luego se usaron con mayúscula en la primera letra (Edad y Nombre) al hacer las visualizaciones.** ✓

¡Ok!

**¡Correcto!**

La respuesta correcta es:

El error es que las variables *edad* y *nombre* se definieron en minúsculas al hacer la carga, y luego se usaron con mayúscula en la primera letra (*Edad* y *Nombre*) al hacer las visualizaciones.

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

Dado un algoritmo, llamamos *instrucciones primitivas* o *acciones primitivas* a aquellos pasos mínimos del algoritmo que necesariamente debe saber aplicar quien ejecute el algoritmo (por ejemplo, para hacer una suma de dos números de varios dígitos, las operaciones primitivas más básicas son alinear los números hacia la derecha, y sumar números de un dígito).

Suponga que se quiere plantear un algoritmo para dibujar un tablero de ajedrez (sin las fichas... SÓLO el tablero). ¿Cuál de las siguientes opciones describe **mejor** el conjunto de acciones primitivas que sería necesario aplicar?

Seleccione una:

- a.  
{ Dibujar cuadrados, Pintar por dentro un cuadrado con un color dado } ✓
- b.  
{ Dibujar cuadrados (sólo el contorno) }
- c.  
{ Dibujar líneas rectas horizontales, Dibujar líneas rectas verticales }
- d.  
{ Dibujar triángulos (sólo el contorno) }

**¡Correcto!**

La respuesta correcta es:

{ Dibujar cuadrados, Pintar por dentro un cuadrado con un color dado }

**Pregunta 13**

Correcta

Puntúa 1 sobre 1

¿Cuáles son los motivos por los cuales una persona que sabe resolver un problema, querría programar y usar una computadora para resolverlo?

Seleccione una:

- a.

No hay motivos para que lo haga: Si sabe resolver el problema, no necesita una computadora y no hay motivo para usarla.

- b.

Porque al programar una computadora, tendrá la garantía de una solución correcta.

- c.

Porque sólo programando una computadora obtendrá soluciones numéricamente precisas y sin errores ni pérdida de precisión por valores decimales.

- d.

Porque al programar una computadora para resolver el problema, ganará tiempo y ahorrará esfuerzo en el futuro: la computadora puede obtener las soluciones muy rápidamente, y con precisión. ✓

¡Ok!

¡Correcto!

La respuesta correcta es:

Porque al programar una computadora para resolver el problema, ganará tiempo y ahorrará esfuerzo en el futuro: la computadora puede obtener las soluciones muy rápidamente, y con precisión.

**Pregunta 14**

Correcta

Puntúa 1 sobre 1

¿Qué relación existe entre los conceptos de algoritmo y programa?

Seleccione una:

- a.

Son exactamente lo mismo.

- b.

Un programa es un algoritmo que sólo puede ser interpretado por una persona.

- c.

Un programa es un algoritmo que puede ser interpretado y ejecutado por un computador. ✓

¡Ok!

- d.

Ninguna relación.

¡Correcto!

La respuesta correcta es:

Un programa es un algoritmo que puede ser interpretado y ejecutado por un computador.

**Pregunta 15**

Correcta

Puntúa 2 sobre 2

¿Hay algún inconveniente en el siguiente script elemental de Python? (Suponga que no hay otras instrucciones previas al script mostrado)

```
a = 10  
print(a)  
a = 'sol'  
print(a)  
a = True  
print(A)
```

Seleccione una:

- a.  
No hay ningún problema.
- b.  
Producirá un error al intentar ejecutar la tercera línea: `a = 'sol'`
- c.  
Producirá un error al intentar ejecutar la quinta línea: `a = True`
- d.  
Producirá un error al intentar ejecutar la última línea: `print(A)` ✓

¡Ok! Exactamente... la variable `A` que se intenta mostrar en la instrucción `print()` de la última línea, no existe. Python es case sensitive: la variable que existe es `a`.

¡Correcto!

La respuesta correcta es:

Producirá un error al intentar ejecutar la última línea: `print(A)`

**Pregunta 16**

Correcta

Puntúa 2 sobre 2

¿Qué valor queda valiendo la variable `x`, luego de la siguiente secuencia de instrucciones en Python 3?

```
p = 2  
x = 20  
x = p * 8  
x = int(input('Ingrese un número entero: '))  
x = 17 + p  
x = p + 1
```

Seleccione una:

- a.  
No se puede saber cuál será el valor final de `x`: depende del valor cargado por teclado en la cuarta línea.
  - b.  
20
  - c.  
19
  - d.  
3 ✓
- ¡Ok!

¡Correcto!

La respuesta correcta es:

3

**Pregunta 17**

Correcta

Puntúa 2 sobre 2

¿Qué efecto produce el siguiente script en Python?

```
b = 20
B = None
y = b * 3
print("Valor de y: ", y)
print("Valor de b: ", B)
```

Seleccione una:

 a.Produce un error al intentar ejecutar la última línea (la variable *B* no está definida). b.Ejecuta sin problemas: queda *b* = 20, *y* = 60, *B* = *None*. ✓¡Ok! Las variables *b* y *B* son diferentes, y no hay problema en asignar el valor *None* a una variable. c.Produce un error al intentar ejecutar la tercera línea: *y* = *b* \* 3 (no se puede usar el valor *None* como numérico) d.Ejecuta sin problemas: queda *b* = *None*, *y* = *None*, *B* = *None*.

¡Correcto!

La respuesta correcta es:

Ejecuta sin problemas: queda *b* = 20, *y* = 60, *B* = *None*.**Pregunta 18**

Correcta

Puntúa 2 sobre 2

¿Hay algún error en la siguiente secuencia de instrucciones en Python?

```
b = None
a = b + 1
b = 1
```

Seleccione una:

 a.La variable *b* está definida, pero con el valor *None* cuando se ejecuta la segunda línea. La suma no puede ejecutarse y lanza un error. ✓

¡Ok!

 b.

No hay error alguno.

 c.

El signo = usado en las tres instrucciones no es un operador válido en Python.

 d.La constante *None* no tiene ningún significado y no existe en Python. Lanza un error en la primera línea.

¡Correcto!

La respuesta correcta es:

La variable *b* está definida, pero con el valor *None* cuando se ejecuta la segunda línea. La suma no puede ejecutarse y lanza un error.

**Pregunta 19**

Correcta

Puntúa 2 sobre 2

Suponga que está trabajando directamente con el editor del *shell de Python* (por ejemplo, a través del *IDLE GUI*). ¿Qué efecto producirá el siguiente script? (aclaración: los símbolos "*>>>*" conforman el prompt del IDLE, y **no deben ser escritos por el programador**... sólo escriba las instrucciones que se marcan abajo en color azul):

```
>>>var = 12
>>>var
```

Observación: también suponemos que el programador escribirá el script *línea por línea presionando <Enter>* al final de cada una en el shell, y **NO** que hará "copy & paste" de este bloque en el shell.

Seleccione una:

- a.  
Ejecutará correctamente, pero no mostrará nada en la consola de salida.
- b.  
Mostrará el valor de la variable *var* (un 12) en la consola de salida. ✓

Correcto. Si está trabajando directamente con el editor del shell, no es necesario usar la función *print()* para visualizar el valor de una variable...

- c.  
Provocará un error de ejecución en la segunda línea.
- d.  
Ejecutará correctamente, pero mostrará en consola de salida el valor *None*.

¡Correcto!

La respuesta correcta es:

Mostrará el valor de la variable *var* (un 12) en la consola de salida.

**Pregunta 20**

Correcta

Puntúa 2 sobre 2

¿Cuál es el valor que terminará valiendo la variable *res* luego del siguiente bloque de instrucciones?

```
a = 15
b = 4
res = a // b
```

Respuesta:  ✓

¡Ok! El operador *//* (doble barra) efectivamente calcula el *cociente entero* y los decimales se truncan.

Revise la Ficha 01, página 20, si contestó mal esta pregunta o tiene dudas (incluso si la constestó bien)

La respuesta correcta es: 3

**Pregunta 21**

Correcta

Puntúa 2 sobre 2

¿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:  ✓

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

Revise la Ficha 01, página 20, si contestó mal esta pregunta o tiene dudas (incluso si la constestó bien)

La respuesta correcta es: 2

**Pregunta 22**

Correcta

Puntúa 2 sobre 2

¿Qué valor queda valiendo la variable **a** luego de la siguiente secuencia de instrucciones en Python?

```
a = 5
b = 3
a = b
```

Respuesta: 3



¡Correcto!

Si contestó mal esta pregunta, revise la Ficha 01, página 16 y todo el contexto de esa sección.

La respuesta correcta es: 3

**Pregunta 23**

Correcta

Puntúa 1 sobre 1

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

¿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'

**Pregunta 24**

Correcta

Puntúa 1 sobre 1

¿Es posible que la misma persona que diseña un algoritmo sea también quien ejecute ese algoritmo?

Seleccione una:

- Verdadero ✓
- Falso

¡Correcto!

Revise la Ficha 01, página 4 y siguientes si le quedaron dudas.

La respuesta correcta es 'Verdadero'

**Pregunta 25**

Correcta

Puntúa 1 sobre 1

¿Puede decirse que un proceso planteado para que tenga un comienzo en un momento dado pero de tal forma de no detenerse jamás, *es un algoritmo*?

Seleccione una:

- Verdadero
- Falso ✓

¡Correcto!

Revise la Ficha 01, páginas 5 y 6 si le quedaron dudas.

La respuesta correcta es 'Falso'

**Pregunta 26**

Correcta

Puntúa 1 sobre 1

Suponga que se le pide desarrollar un programa que muestre en pantalla todos y cada uno de los números naturales (todos los enteros positivos) ¿Puede hacerse un programa así?

Seleccione una:

- Verdadero
- Falso ✓

¡Correcto!

Revise la Ficha 01, páginas 5 y 6 si le quedaron dudas.

La respuesta correcta es 'Falso'

**Comenzado el** sábado, 7 de abril de 2018, 19:35

**Estado** Finalizado

**Finalizado en** domingo, 15 de abril de 2018, 16:56

**Tiempo empleado** 7 días 21 horas

**Puntos** 23/23

**Calificación** 10 de 10 (100%)

**Pregunta 1**

Correcta

Puntúa 2 sobre 2

¿Cuál de las siguientes expresiones Python **NO** calcula la *raíz cuarta* del valor contenido en la variable *a*? (Suponga que *a* contiene un número positivo)

Seleccione una:

- a. `r1 = a**0.25`
- b. `r1 = a**(1/4)`
- c. `r1 = pow(a, (1/4))`
- d. `r1 = pow(0.25, a)` ✓

¡Ok! Efectivamente, la expresión está calculando el valor de  $(0.25)^a$  en lugar de  $a^{0.25}$  que es lo que se pedía.

¡Correcto!

La respuesta correcta es: `r1 = pow(0.25, a)`

**Pregunta 2**

Correcta

Puntúa 2 sobre 2

Suponga que el Departamento de Documentación del Registro Civil cuenta con 7 oficinas numeradas en forma correlativa entre 1 y 7. Cada persona que llega a realizar un trámite debe ser enviada a una de las siete oficinas y para determinar el número de la oficina se usa como dato el número de *dni* de la persona. Suponiendo que el número de *dni* está almacenado correctamente en la variable *dni*, ¿cuál de las siguientes expresiones calculará correctamente el número de la oficina donde debe enviarse a cada persona?

Seleccione una:

a.

`oficina = dni % (7 + 1)`

b.

`oficina = dni // 7 + 1`

c.

`oficina = dni % 7`

d.

`oficina = dni % 7 + 1` ✓

¡Ok! Efectivamente, el cálculo `dni % 7` entrega un valor que siempre estará en el intervalo  $[0, 6]$ ... por lo tanto, si se suma 1 se ajusta el resultado para que calce en el intervalo  $[1, 7]$ .

¡Correcto!

La respuesta correcta es:

`oficina = dni % 7 + 1`

**Pregunta 3**

Correcta

Puntúa 3 sobre 3

¿Cuál de los siguientes conjuntos **NO** es la caracterización de una clase de congruencia (módulo 4)? (Recuerde que se denota como  $Z$  al conjunto de los números enteros)

Seleccione una:

- a.

$$\{ 4*k + 1 \mid (\forall k \in Z) \}$$

- b.

$$\{ 5*k + 1 \mid (\forall k \in Z) \}$$



¡Ok! Efectivamente, los números que caracterizan a este conjunto son de la forma  $5*k + 1$ , lo cual indica que todos ellos dejan un resto de 1 pero al dividir por 5 (y no necesariamente por 4). Por lo tanto, el conjunto mostrado es una clase de congruencia (**módulo 5**) [de hecho,  $Z_5$ ] Y NO (módulo 4).

- c.

$$\{ 4*k \mid (\forall k \in Z) \}$$

- d.

$$\{ 4*k + 3 \mid (\forall k \in Z) \}$$

¡Correcto!

La respuesta correcta es:

$$\{ 5*k + 1 \mid (\forall k \in Z) \}$$

**Pregunta 4**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es cierta respecto de un *diagrama de flujo*?

Seleccione una:

- a.

Se hace un diagrama de flujo para cada lenguaje en que se vaya a programar, aunque el problema sea siempre el mismo.

- b.

Es un gráfico que permite ver claramente la lógica de un algoritmo, sin entrar en los detalles de la sintaxis de un lenguaje de programación. 

¡OK!

- c.

Un diagrama es un gráfico que los profesores inventaron para torturar a los sufridos y nunca bien comprendidos alumnos.

- d.

En un diagrama de flujo deben ponerse hasta los detalles mínimos: colores usados en la pantalla, mensajes aclaratorios en pantalla, comas y símbolos específicos de la sintaxis de un lenguaje, etc.

¡Correcto!

La respuesta correcta es:

Es un gráfico que permite ver claramente la lógica de un algoritmo, sin entrar en los detalles de la sintaxis de un lenguaje de programación.

**Pregunta 5**

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **falsa** respecto de la técnica de **pseudocódigo** para representación de algoritmos?

Seleccione una:

a.

El pseudocódigo está pensado para ser leído e interpretado por una persona, y no por una computadora.

b.

El planteo de un esquema de pseudocódigo se realiza siempre en base a reglas y estándares estrictos que los programadores deben conocer y respetar. ✓

¡Ok! Recuerde: se pidió indicar cuál de las consignas es FALSA... y esto es efectivamente falso en cuanto al planteo de pseudocódigos.

c.

El pseudocódigo puede basarse (con menor o mayor rigor) en la estructura general de un lenguaje particular, y en este caso se designa como un *pseudocódigo estructurado*.

d.

El programador es quien decide la forma, la profundidad y el nivel de detalle expresado en la lógica y en la estructura de un algoritmo.

¡Correcto!

La respuesta correcta es:

El planteo de un esquema de pseudocódigo se realiza siempre en base a reglas y estándares estrictos que los programadores deben conocer y respetar.

**Pregunta 6**

Correcta

Puntúa 1 sobre 1

¿Qué es en programación una *Estructura Secuencial de Instrucciones*?

Seleccione una:

a.

Una tupla compuesta por  $n$  variables del mismo tipo.

b.

Un bloque de comentarios de texto incluido en un programa.

c.

Un bloque de instrucciones simples (asignaciones, visualizaciones, lecturas) escritas una debajo de la otra y ejecutadas en el orden que aparecen.



Ok.

d.

Una forma de organizar un conjunto de  $n$  datos para facilitar su acceso desde un programa.

¡Correcto!

La respuesta correcta es:

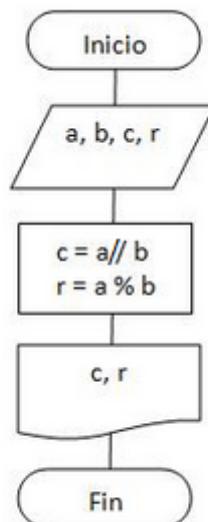
Un bloque de instrucciones simples (asignaciones, visualizaciones, lecturas) escritas una debajo de la otra y ejecutadas en el orden que aparecen.

**Pregunta 7**

Correcta

Puntúa 1 sobre 1

Suponga que se desea desarrollar un programa que cargue dos números, y muestre el cociente entero y resto de la división entre esos dos números. A continuación se muestra el diagrama de flujo propuesto y el programa en Python. ¿Está bien planteado el diagrama de flujo?



```
# script en Python...
a = int(input('A: '))
b = int(input('B: '))

c = a // b
d = a % b

print('Cociente:', c)
print('Resto:', d)
```

Seleccione una:

- a.

El diagrama está mal planteado: en el símbolo de carga por teclado (el paralelogramo) está indicando la **carga** de las cuatro variables, cuando solo deben cargarse dos. ✓

¡Ok!

- b.

El diagrama está mal planteado (y también el programa): debió usarse una condición para verificar primero si el divisor (*b*) es menor o igual al dividendo (*a*).

- c.

El diagrama está mal planteado: el símbolo usado al final para indicar visualización de resultados, debió ser un rectángulo.

- d.

Sí. El diagrama está correctamente planteado.

¡Correcto!

La respuesta correcta es:

El diagrama está mal planteado: en el símbolo de carga por teclado (el paralelogramo) está indicando la **carga** de las cuatro variables, cuando solo deben cargarse dos.

**Pregunta 8**

Correcta

Puntúa 2 sobre 2

Suponga que las variables  $a$ ,  $b$  y  $c$  están correctamente asignadas en forma previa, y considere la siguiente expresión en Python:

```
d = a + b + c // a - b - c
```

¿Cuál de las siguientes es equivalente a la expresión anterior (cuál de ellas obtiene idéntico resultado para la variable  $d$ )?

Seleccione una:

1.

$d = a + b + c // (a - b - c)$

2.

$d = a + b + (c // a - b - c)$  ✓

¡Ok!

3.

$d = (a + b + c) // (a - b - c)$

4.

$d = (a + b + c) // a - b - c$

¡Correcto!

La respuesta correcta es:

$d = a + b + (c // a - b - c)$

**Pregunta 9**

Correcta

Puntúa 2 sobre 2

¿Hay algún inconveniente en el script Python que sigue?

```
a = int(input('A: '))
b = int(input('B: '))
c = a + b
print('Suma:', c)
```

Seleccione una:

- a.  
Sí. Al cargar los datos, estos se están ingresando y asignando como cadenas de caracteres, de modo que al hacer la suma el resultado será la concatenación de las cadenas en lugar de la suma de los números esperados.
- b.  
Sí. Están mal usadas las funciones input() e int() (no pueden combinarse en la forma mostrada en el script).
- c.  
No. No hay ningún problema.
- d.  
Sí. Está mal indentada la última línea, y provocará un error de intérprete. ✓

Ok.

¡Correcto!

La respuesta correcta es:

Sí. Está mal indentada la última línea, y provocará un error de intérprete.

**Pregunta 10**

Correcta

Puntúa 2 sobre 2

¿Qué hace el siguiente script en Python?

```
__author__ = 'Cátedra de AED'

c1 = float(input('Ingrese el primer valor: '))
c2 = float(input('Ingrese el segundo valor: '))
c3 = float(input('Ingrese el tercer valor: '))

res = (c1 + c2 + c3) / 3
print('Resultado:', res)
```

Seleccione una:

- a.  
Calcula y muestra el cociente entre el valor c3 y el número 3.
- b.  
Calcula y muestra el porcentaje que el valor c1 representa sobre el total  $c1 + c2 + c3$ .
- c.  
Calcula y muestra el promedio real de los valores c1, c2 y c3. ✓  
¡OK!
- d.  
Calcula y muestra la suma entre los valores c1, c2 y c3.

¡Correcto!

La respuesta correcta es:

Calcula y muestra el promedio real de los valores c1, c2 y c3.

**Pregunta 11**

Correcta

Puntúa 2 sobre 2

¿Cuáles de las siguientes son propiedades básicas del resto de una división (y por lo tanto, aplicables al *operador resto o módulo* en un lenguaje de programación)?

**Observación:** note que MAS DE UNA respuesta puede ser correcta. Marque TODAS las que considere válidas.

Seleccione una o más de una:

- a.

El resto de dividir un número entero positivo  $x$  por otro entero positivo  $n$ , puede ser un número mayor a  $n$ .

- b.

Si se divide un número entero positivo  $x$  por otro número entero positivo  $n$ , los posibles restos son todos los números en el intervalo  $[0, n-1]$  (y serían entonces,  $n$  posibles valores distintos). ✓

¡Ok! Se deduce de la propia definición del resto de una división entre números enteros positivos.

- c.

El resto de dividir un número  $x$  por otro  $n$ , puede ser igual al número  $x$ . ✓

¡Ok! Esto ocurrirá cada vez que  $x < n$ ... Por ejemplo: si  $x = 5$  y  $n = 7$  entonces el cociente es cero... y el resto será 5...

- d.

Si el resto de dividir un número  $x$  por otro  $n$  es cero, entonces  $x$  es múltiplo de  $n$  (o lo que es lo mismo,  $x$  es divisible por  $n$ ). ✓

¡Ok!

¡Correcto!

Las respuestas correctas son:

Si se divide un número entero positivo  $x$  por otro número entero positivo  $n$ , los posibles restos son todos los números en el intervalo  $[0, n-1]$  (y serían entonces,  $n$  posibles valores distintos).,

Si el resto de dividir un número  $x$  por otro  $n$  es cero, entonces  $x$  es múltiplo de  $n$  (o lo que es lo mismo,  $x$  es divisible por  $n$ ).,

El resto de dividir un número  $x$  por otro  $n$ , puede ser igual al número  $x$ .

**Pregunta 12**

Correcta

Puntúa 1 sobre 1

Sabemos que un *IDE* es un programa que provee herramientas para editar, depurar y ejecutar con sencillez y eficiencia un programa desarrollado en algún lenguaje de programación. El *IDE* que usaremos a lo largo del curso es el *PyCharm Edu*. Concretamente, ¿qué significa la sigla *IDE*?

Seleccione una:

- a.  
Integrated Development Engine (Motor Integrado de Desarrollo)
- b.  
Integrated Development Environment (Entorno Integrado de Desarrollo)  

- c.  
Integrated Database Environment (Entorno Integrado de Bases de Datos)
- d.  
Integrated Database Engine (Motor Integrado de Bases de Datos)

¡Correcto!

La respuesta correcta es:

Integrated Development Environment (Entorno Integrado de Desarrollo)

**Pregunta 13**

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes símbolos NO representa un *proceso* en un *diagrama de flujo*?

Seleccione una:

a.



?

b.

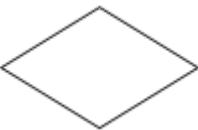


¡Ok! Efectivamente, este símbolo representa una operación de carga de datos y no un proceso de transformación de datos en resultados.

c.



d.



¡Correcto!

La respuesta correcta es:



**Pregunta 14**

Correcta

Puntúa 2 sobre 2

¿Cuál es el valor final de la variable `res`, luego de aplicar la siguiente secuencia de instrucciones en Python? (Es recomendable que primero intente ejecutar este script y luego conteste a esta pregunta):

```
a = 20  
b = 6  
res = ((a // b) * 4) % 7
```

Respuesta: 5



¡Correcto!

Si contestó mal esta pregunta, revise la Ficha 02, página 42 y siguientes, más la sección de Temas Avanzados de la misma Ficha 2. Y aún mejor, haga un programa en Java que realice el cálculo pedido y lo muestre por consola, para asegurarse...

La respuesta correcta es: 5