NOMBRE APELLIDOS

Examen de PECT1 Grado de Criminalística :

NORMAS: No utilice bolígrafo verde. No tenga ninguna hoja sobre la mesa sin su nombre. Conteste concretamente a lo que se le pregunta, todo lo que se indique no relevante, restará puntos (incluso afectando a la calificación de otras preguntas).

T1. [1 punto] Escriba la expresión más simple posible equivalente (es decir que haga lo mismo) a la siguiente. Ponga datos numéricos de la mejora conseguida. Justifique la respuesta.

```
if not(s or not s) and not(not r or not not r): print('Hola')
```

T2. El siguiente código devuelve la suma de los elementos que se encuentran antes del primer impar, de un array de tamaño MAX, (o la suma de todos sus elementos, si no hubiera ningún impar).

```
# Definición subalgoritmos
def acumular(mivector, n):
    ''' array -> int
    OBJ: Suma los primeros números hasta que encuentra un impar'''
    for i in range(n):
        if mivector[i] % 2 == 1:  break
        else:  suma = suma + mivector[i]
    return suma

# Cuerpo del programa
# v es un vector de tamaño Max que ya está cargado de números enteros
# MAX es una constante definida previamente
…
suma = 0
suma = acumular(v, MAX)
print('La suma es:', str(suma))
```

a) [0,6] Indique qué instrucciones (de las presentes en el código) están desaconsejadas en la programación modular, explicando el motivo por el qué están desaconsejadas.

b) [0,6] Para qué usa el sangrado Python, ¿Qué diferencias tiene con el empleo el sangrado en otros lenguajes (como Pascal o C)? Indique los errores de sangrado en el código anterior.

NOMBRE APELLIDOS

c) [0,6] ¿Cómo afecta a la elección de los identificadores el hecho de que lo nombrado sea un recurso de una biblioteca o, por el contrario, sea una variable local de un subprograma?

d) [0,6] Enuncie la **parte del convenio de identificadores de subprogramas** que es aplicable en el código mostrado. Justifique las ventajas e inconvenientes de esa parte del convenio e indique el nombre adecuado para este subprograma si forma parte de una biblioteca.

e) [0,6] Discuta la adecuación de los parámetros diseñados en el código anterior. Ventajas, inconvenientes, de cada uno, alternativas, etc.

f) [1] Indique cómo afectan a la calidad del software las consultas y modificaciones globales ¿Cuándo está permitido hacerlas? ¿Qué se busca al (cuales son las ventajas de) permitirlo? Indique las situaciones relacionadas con este concepto en el código anterior.

NOMBRE APELLIDOS

g) [0,6] Los caracteres “””, “’’, ‘,”, y # pueden jugar papeles equivalentes en un programa Python, pero de hecho hay un convenio que aconseja reservar cada uno para diferentes situaciones. Indique cuál es ese convenio y explique qué beneficios tiene aplicarlo.

h) [0,6] Explique la diferencia entre Python y lenguajes de programación como C, Pascal o Java, respecto a la primera línea de documentación que nuestro convenio recoge para los subprogramas Python. Indique cómo lo gestionan esos otros lenguajes y porqué en Python es diferente.

i) [0,6] Algunos lenguajes como C hacen que todos sus subprogramas devuelvan al menos un código de error. Diseñe un código que muestre cual es el comportamiento a este respecto del print() de Python.

j) [0,6] ¿De qué tipo han de ser los parámetros de entrada a una llamada a print de Python?. Especifique la funcionalidad de dos parámetros especiales de dicho procedimiento.

k) [0,8] Justifique cómo optimizaría el código mostrado al principio de este ejercicio para que tarde menos.

l) [1]Teniendo en cuenta todo lo anterior, reescriba el mejor código para el subprograma presentado (eficacia, legibilidad_mantenibilidad, reusabilidad y, por supuesto, eficiencia).

NOMBRE APELLIDOS

T3 [0,8] En los errores frecuentes del for está escrito este código, donde cont, ini, y fin están previamente inicializados. ¿Qué hace el código (ponga ejemplos) ? Porqué está desaconsejado.

```
for cont in range(ini,fin):
    fin=fin+2
    print('tocado, ini, fin')
print('fin fuera del bucle',fin)
```

NOMBRE APELLIDOS
PROPUESTA DE SOLUCION

T1. [1 punto] Escriba la expresión más simple posible equivalente (es decir que haga lo mismo) a la siguiente. Ponga datos numéricos de la mejora conseguida. Justifique la respuesta.

```
if not(s or not s) and not(not r or not not r): print('Hola')
```

La instrucción se puede eliminar completamente del programa, pues nunca se cumplirá. "`s or not s`" es siempre cierta y, por tanto, su negación es falsa, y por lo tanto, el and también lo es.

El número de operaciones que contiene la expresión es 9 pero, teniendo en cuenta la evaluación perezosa, se ejecutarán en este orden: segundo not, primer or, primer not y an evaluar el and, decidirá que no continúa, de modo que, eliminando la instrucción, el programa hará lo mismo, pero tardará 4 tiempos de reloj menos.

T2. El siguiente código devuelve la suma de los elementos que se encuentran antes del primer impar, de un array de tamaño MAX, (o la suma de todos sus elementos, si no hubiera ningún impar).

```
# Definición subalgoritmos
def acumular(mivector, n):
    ''' array -> int
    OBJ: Suma los primeros números hasta que encuentra un impar'''
    for i in range(n):
        if mivector[i] % 2 == 1:  break
        else:  suma = suma + mivector[i]
    return suma

# Cuerpo del programa
# v es un vector de tamaño Max que ya está cargado de números enteros
# MAX es una constante definida previamente
…
suma = 0
suma = acumular(v, MAX)
print('La suma es:', str(suma))
```

a) [0,6] Indique qué instrucciones (de las presentes en el código) están desaconsejadas en la programación modular, explicando el motivo por el qué están desaconsejadas.

Desaconsejado break, ya que todo módulo debe tener un único punto de entrada y uno de salida y break es un punto adicional.

b) [0,6] Para qué usa el sangrado Python, ¿qué diferencias tiene el empleo del sangrado con su uso en otros lenguajes (como Pascal o C)? Indique los errores de sangrado en el código anterior.

En Python, el sangrado marca los bloques (cuerpos de código) al interprete. En otros lenguajes el sangrado es transparente al compilador, ya que disponen de indicadores de inicio y fin de bloque. En otros lenguajes, se incluye para aumentar la legibilidad. No hay errores de sangrado en el código.

c) [0,6] ¿Cómo afecta a la elección de los identificadores el hecho de que lo nombrado sea un recurso de una biblioteca o, por el contrario, sea una variable local de un subprograma?

Si el ámbito de una variable/constante/subprograma es pequeño, probablemente en una porción que alcance la vista humana, y posiblemente acompañado de documentación, podemos permitirnos que los nombres sean cortos, ahorrando esfuerzo al programador. Las bibliotecas se construyen para poner recursos a disposición de otros subprogramas y se hará uso de ellas lejos de su documentación. Los nombres deben resultar muy significativos.

NOMBRE APELLIDOS

d) [0,6] Enuncie la **parte del convenio de identificadores de subprogramas** que es aplicable en el código mostrado. Justifique las ventajas e inconvenientes de esa parte del convenio e indique el nombre adecuado para este subprograma si forma parte de una biblioteca.

Este código es una función, ya que devuelve un valor. El nombre de la pieza debe reflejar la semántica del valor que devuelve, por tanto, será un sustantivo. En este caso sumaHastaImpar o suma_hasta_impar.

Me centro en el convenio específico para identificadores de subprogramas, que es lo pedido.En el convenio genérico de identificadores: ha de estar escrito en minúsculas (por el papel que juega en el código) y si es compuesto, bien palabras separadas tal como indicado en el ejemplo, pero realmente, esto no es lo pedido.

e) [0,6] Discuta la adecuación de los parámetros diseñados en el código anterior. Ventajas, inconvenientes, de cada uno, alternativas, etc.
Respecto a los parámetros de entrada: Puesto que voy a sumar elementos del vector, el subprograma necesita recibir el vector, sin embargo, el subprograma es capaz de determinar por sí mismo el tamaño del vector recibido. Es inadecuado pasárselo:

A) porque dificulta el reuso, al hacer innecesariamente compleja su llamada
B) favorece inconsistencias→ disminuye la robustez, ¿Qué ocurre si por error se envía un tamaño mayor del real y solo hay pares? → el programa abortará.
C) Caso de mantener este segundo parámetro, sería indispensable incluir una nueva PRE: que empeora la legibilidad del código.

De otra parte Suma es el parámetro de salida

Convendría reflexionar sobre el consumo de memoria que conlleva la copia de vectores potencialmente grandes, afectando a la eficiencia…pero, puede requerir conocimientos mas profundos

f) [1] Indique cómo afectan a la calidad del software las consultas y modificaciones globales ¿Cuándo está permitido hacerlas? ¿Qué se busca al (cuales son las ventajas de) permitirlo? Indique las situaciones relacionadas con este concepto en el código anterior.

Las modificaciones globales producen efectos laterales, hasta el punto de que en Python hay que avisar expresamente al compilador que se desea hacerlo. Las consultas globales están permitidas a constantes. Ventaja evita que pasemos más parámetros a los subprogramas en cada llamada (y ahorra algo de tiempo y memoria, especialmente importante si el parámetro de entrada es "pesado" y mutable=tupla,namedtupla). Desventaja, hay menor cohesión. Las piezas de código tienen entre si mayor acoplamiento Es decir, el subp aborta si no está declarada antes la constante.--> hay que indicarlo en la PRE.

En el código pretende hacer una modificación global, por lo que abortará. La instrucción suma=0 hay que meterla dentro del subprograma.

g) [0,6] Los caracteres """, ''', ',", y # pueden jugar papeles equivalentes en un programa Python, pero de hecho hay un convenio que aconseja reservar cada uno para diferentes situaciones. Indique cuál es ese convenio y explique qué beneficios tiene aplicarlo.

NOMBRE APELLIDOS

‘ delimita constantes tipo cadena
“ solo suele usarse si la constante cadena ha de contener comillas
sirve para documentación puntual
“”” sirve para documentación mas extensa
‘’’ para desactivar código.
Cuando en la fase de pruebas, necesitemos desactivar código que está documentado, se mezclarían los marcadores, no consiguiendo fácilmente en objetivo deseado

h) [0,6] Explique la diferencia entre Python y lenguajes de programación como C, Pascal o Java, respecto a la primera línea de documentación que nuestro convenio recoge para los subprogramas Python. Indique cómo lo gestionan esos otros lenguajes y porqué en Python es diferente.

Los lenguajes con un fuerte control de tipo como los mencionados, requieren que se especifique el tipo de los argumentos de entrada/salida/entrada+salida como parte del propio lenguaje compilable, por tanto, no hay que documentarlos.

i) [0,6] Algunos lenguajes como C hacen que todos sus subprogramas devuelvan al menos un código de error. Diseñe un código que muestre cual es el comportamiento a este respecto del print() de Python.
print(print('hola')) imprimirá Hola none

m) [0,6] ¿De qué tipo han de ser los parámetros de entrada a una llamada a print de Python?. Especifique la funcionalidad de dos parámetros especiales de dicho procedimiento.

j) El procedimiento print imprime cualquier tipo de datos lícito en Python. Tiene dos parámetros especiales end=(caracteres que añade al final del print)y sep= (caracteres de separación entre parámetros)

n) [0,8] Justifique cómo optimizaría el código mostrado al principio de este ejercicio para que tarde menos.

A) La función range genera una secuencia, que en este caso nos va a servir para recorrer un vector existente, podemos recorrer directamente el vector.
Pero, en cualquier caso, hemos de sustituir este for por un while, para evitar el break, aumentando la estructuración y por tanto la legibilidad (aunque esto no disminuya el tiempo que tarda)

B) Suma+=lo_que_sumo evita buscar dos veces el sumador en la tabla de símbolos

k) [1]Teniendo en cuenta todo lo anterior, reescriba el mejor código para el subprograma presentado (eficacia, legibilidad_mantenibilidad, reusabilidad y, por supuesto, eficiencia).

```
def suma_hasta_impar (miVector):
   """sec(int)-->int
   OBJ: Suma los primeros números hasta que encuentra un impar"""
   lon=len(miVector)
   suma=0
   pos=0
   while pos<lon and miVector[pos]%2==0:
      suma+=miVector[pos]
      pos+=1
   return suma
v=(2,4,6,1,2)
```

NOMBRE APELLIDOS

```
print('La suma hasta impar es:', suma_hasta_impar(v))
```

T3 [0,8] En los errores frecuentes del for está escrito este código, donde cont, ini, y fin están previamente inicializados. ¿Qué hace el código (ponga ejemplos) ? ¿Por qué está desaconsejado?

```
for cont in range(ini,fin):
    fin=fin+2
    print('tocado, ini, fin')
print('fin fuera del bucle',fin)
```

Es mala idea modificar los controles de un for dentro del propio for, porque dificulta la legibilidad, consume operaciones innecesarias, pues siempre se puede calcular fuera del bucle. Incluso en distintos lenguajes podría variar el comportamiento. Siendo f, e i los valores de fin e ini antes de entrar al bucle, n Python imprimirá “tocado, ini,fin” f-i-1 veces. Y 1 vez “fuera del bucle f+(f-i-1)*2”.

Entradas	Pantalla
ini=1 fin=2	tocado, ini, fin fin fuera del bucle
ini=4 fin=6	tocado, ini, fin tocado, ini, fin fin fuera del bucle 10