Universidad de Alcalá

Fundamentos de la programación, Curso 2020-21
PEC2-TEORÍA

Nombre:
Puesto:
Titulación:

1. Indica si las siguientes afirmaciones son verdaderas o falsas. **[+0,25p. acierto, -0,1p. fallo]**
   - **a.** Toda subprograma en Python debe tener al menos una cláusula `return` [V / F]
   - **b.** La sentencia a=b es equivalente a esta otra: `a = b[0:len(a)]` [V / F]
   - **c.** Al ser inmutables los strings no es posible hacer cosas como `texto[0]='X'` **[V / F]**
   - **d.** Si llamamos a una función de 2 argumentos usando 3 argumentos, el código provocará un error en ejecución **[V / F]**

SOLUCION:

a – Falso: puede no haber cláusula return
b – Falso: la primera pone la referencia b a apuntar al mismo objeto que a mientras que la segunda hace una copia de la secuencia a y pone a apuntar b a esa copia (distintos objetos tras la ejecución del código).
c – Verdadero
d – Falso, se produce un error en compilación/traducción

2. Indica el resultado del siguiente código: **[1,5 p.]**

```
def funcion_misteriosa(mi_lista):
  """ lista -> lista
      OBJ: *** no se muestra ***
  """
  a = []
  for elemento in mi_lista:
    if type(elemento) == list:
      a += funcion_misteriosa(elemento)
    else:
      a += elemento
  return a

print(funcion_misteriosa(['a', ['b', ['c', ['d']], 'e'], 'f']))
```

SOLUCION PROPUESTA

```
['a', 'b', 'c', 'd', 'e', 'f']
```

3. Completa el siguiente código para generar una lista con tantas tuplas como elementos haya en `lista_b`. Cada tupla contiene el número de la `lista_b` y su posición en `mi_lista`. Por ejemplo: para el código que se muestra la salida en pantalla sería [(6, 2), (4, 8), (6, 2), (1, 3), (2, 7), (2, 7)]. Hay garantía de que todo elemento de `lista_b` está en `mi_lista` y esta no contiene valores repetidos. Puedes usar funciones de secuencia. **[1 p.]**

```
lista_b = [6, 4, 6, 1, 2, 2]
mi_lista = [9, 3, 6, 1, 5, 0, 8, 2, 4, 7]
resultado = []
_____________________________:
  result.append(_____________________________)

print(resultado) # [(6, 2), (4, 8), (6, 2), (1, 3), (2, 7), (2, 7)]
```

SOLUCION PROPUESTA

```
for i in lista_b:
    resultado.append((i, mi_lista.index(i)))
```

4. Indica la salida del siguiente código: **[1 p.]**

```
def otro_misterio(mi_lista, n, cuanto):
  """ lista, int, int -> None
  """
  for i in range(len(mi_lista)):
    mi_lista [n][i] += cuanto
  print(mi_lista [n])

sublista = [1,2,3,4,5]
mi_lista = []
for i in range(5):
  mi_lista.append(sublista)
otro_misterio(mi_lista,0, 3)
print(mi_lista)
```

SOLUCION PROPUESTA

```
[4, 5, 6, 7, 8]
[[4, 5, 6, 7, 8], [4, 5, 6, 7, 8], [4, 5, 6, 7, 8], [4, 5, 6, 7, 8], [4, 5, 6, 7, 8]]
```

5. Completa el siguiente código para que en un texto en inglés formado solo por palabras (sin signos de puntuación, exclamaciones, etc.) se sustituya la primera aparición de la subcadena entre las palabras 'not' y 'poor' (ambas incluidas) por la subcadena 'good'. Si no existe una subcadena 'not…poor' la función devuelve el texto original sin modificar. **[2 p.]**

   Ejemplo 1: 'The film is **not that poor**' se transformaría en 'The film is **good**'.
   Ejemplo 2: 'The film is poor, not recommended at all' no se transformaría.

```
def not_poor(texto):
   """ str -> str
   """
    s_not =  _____________________ # encuentra la primera aparición de 'not'
    if s_not != -1:
        s_poor = s_not + texto[s_not:].find('poor')
        if ___________________ :
            texto = _____________________
    return texto
```

SOLUCIONES PROPUESTAS

```
s_not = texto.find('not')
if s_poor > s_not
texto = texto.replace(texto [s_not:(s_poor+4)], 'good')
```

```
s_not = texto.find('not')
if s_poor > s_not
texto = texto[:s_not]+ 'good' + texto[s_poor+4:]
```

6. La ferretería del barrio quiere informatizar el inventario y te ha contratado para diseñar la estructura de datos que permita almacenar la información de la siguiente tabla:

CODIGO	PRODUCTO	PRECIO	STOCK
REF_01	Destornillador	12,50	10
REF_02	Taladro	89,00	5

REF_03	Radial	150,00	8
REF_04	Llave inglesa	9,50	10

a) Escribe el código Python necesario para almacenar los datos de la tabla. **[0,5 p.]**

SOLUCION PROPUESTA

```
Lista de diccionarios:

[ {  'cod':'ref_01',
     'producto': 'destornillador',
     'stock': 10,
     'precio': 12.50
  },
  {  'cod':'ref_02',
     'producto': 'taladro',
     'stock': 5,
     'precio': 89
  } # , etc.
]

También es correcta una implementación con una lista de registros
implementada con named_tuples, record_class u otras alternativas
```

b) Completa el siguiente código para ordenar el inventario de forma descendente por stock y si dos elementos tienen el mismo stock, por precio. **[1,5 p.]**

```
def inventario_ordenado(inventario):
  < ***** documentacion (docstring) ***** >
  ordenado = inventario
  for i in range(len(ordenado)):
    for j in range(0, len(ordenado)-i-1):
      if < ***** condicion ***** >
        < ***** sentencia(s) ***** >
  return ordenado
```

SOLUCION PROPUESTA

<documentación (docstring)>

```
list -> list
OBJ: ordena el inventario de forma descendente por stock y a igual stock por precio
```

<condicion>

```
if ordenado[j]['stock'] < ordenado[j+1]['stock'] \
   or ordenado[j]['stock'] == ordenado[j+1]['stock'] \
   and ordenado[j]['precio'] < ordenado[j+1]['precio']:
       ordenado[j], ordenado[j+1] = ordenado[j+1], ordenado[j]
```

<sentencia(s)>

```
ordenado[j], ordenado[j+1] = ordenado[j+1], ordenado[j]
```

c) ¿El subprograma anterior modifica la variable inventario? Justifica tu respuesta. **[0,5 p.]**

SOLUCION PROPUESTA

La variable inventario se modifica ya que no se está haciendo una copia completa de la lista al principio de la función sino creando una nueva variable para manejar la misma lista.

7. El siguiente código pretende sumar una serie finita de los cuadrados de números reales (con incrementos de 0,1) comprendidos entre dos enteros INICIO y FIN, pero no funciona correctamente y entra en un bucle infinito. Explica el motivo y propón una solución. **[1 p.]**

```
INC = 0.1
actual = INICIO
suma = 0.0
while actual != FIN:
    suma+=actual**2
    actual+=INC
print(suma)
```

SOLUCION PROPUESTA

El problema radica en la inexactitud que provoca la forma de almacenamiento interno de los reales, la cual provoca errores de redondeo. Podría resolverse cambiando la condición del `while`, así: `while actual < FIN`