Contenido creado por: Isabel Maniega

Question 1 (Data Aggregates)

What is the expected output of the following code?

```
x = [0, 1, 2]
x.insert(0, 1)
del x[1]
print(sum(x))
```

A. `5`

B. `4`

C. `3`

D. `2`

Solution 1

In [ ]:
```
x = [0, 1, 2]     #   0 1 2
x.insert(0, 1)    # 1 0 1 2      (insertas en indice 0 un "1")
del x[1]          # 1   1 2  ==> 1 1 2
print(sum(x))     # 4            (1+1+2)
```

In [ ]:
```
# Solución
# B
```

Question 2 (Operators)

You develop a Python application for your company.

You have the following code.

```
def main(a, b, c, d):
    value = a + b * c - d
    return value
```

Which of the following expressions is equivalent to the expression in the function?

A. `(a + (b * c) - d)`

B. None of the above

C. `(a + b) * (c - d)`

D. `a + ((b * c)) - d`

Solution 2

In [ ]:
```
def main(a, b, c, d):
    value = a + b * c - d
    return value
```

In [ ]:
```
main(1,2,3,4)
```

In [ ]:
```
# (a + (b * c) - d)
(1 + (2 * 3) - 4)
```

In [ ]:
```
# Solución
# A
```

Question 3 (Error Handling)

What will happen when you attempt to run the following code?

```
print(Hello, World!)
```

A. The code will raise the `SyntaxError` exception

B. The code will print `Hello, World!` to the console

C. The code will raise the `TypeError` exception

D. The code will raise the `ValueError` exception

E. The code will raise the `AttributeError` exception

Solution 3

In [ ]:
```
print(35)
```

In [ ]:
```
# print(Hello, World!)

# SyntaxError: invalid syntax
```

In [ ]:
```
print("Hello, World!")
```

In [ ]:
```
# Solución
# A
```

Question 4 (Functions)

What is the output of the following code snippet?

```
def test(x=1, y=2):
    x = x + y
    y += 1
    print(x, y)

test(2, 1)
```

A. `3 3`

B. `1 3`

C. The code is erroneous

D. `3 2`

E. `2 3`

Solution 4

In [1]:
```
def test(x=1, y=2):
    x = x + y         # x = x + y  -> 3 <= 2 + 1  => x = 3
    y += 1            # y = y + 1  -> 2 <= 1 + 1  => y = 2
    print(x, y)       # 3, 2

test(2, 1)
```

```
3 2
```

In [2]:
```
def test(x=1, y=2):
    print('argumentos de la función: x=1, y=2')
    print('valores en la llamada:    x=2, y=1')
    print("valores reales en este instante:")
    print('x=',x, 'y=',y)
    print("\n")
    x = x + y         # x = x + y  -> 3 <= 2 + 1
    y += 1            # y = y + 1  -> 2 <= 1 + 1
    print(x, y)       # 3, 2

test(2, 1)
```

```
argumentos de la función: x=1, y=2
valores en la llamada:    x=2, y=1
valores reales en este instante:
x= 2 y= 1


3 2
```

In [ ]:
```
# Solución
# D
```

Question 5 (Data Aggregates)

Take a look at the snippet, and choose the true statements:

```
nums = [1, 2, 3]
vals = nums
del vals[1:2]
```

SELECT TWO ANSWERS

A. nums and vals are of the same length

B. nums and vals refer to the same list

C. nums is no longer than vals

D. vals is longer than nums

Solution 5

In [1]:
```
nums = [1, 2, 3]
vals = nums
del vals[1:2]
# añado:
print('vals:', vals)    # borro 2, [1,2]
print('nums:', nums)    # = que vals

# conclusión:
# SI NO HACES UNA COPIA DE LA LISTA,
# TE GUARDA LOS CAMBIOS
```

```
vals: [1, 3]
nums: [1, 3]
```

In [ ]:
```
# Solución
# A y B
```

Question 6 (Data Types)

The value thirty point eleven times ten raised to the power of nine

should be written as:

A. `30.11*10^9`

B. `30.11E9.0`

C. `30E11.9`

D. `30.11E9`

Solution 6

In [2]:
```
30.11E9
```

Out[2]: 30110000000.0

In [ ]:
```
# Solución
# D
```

Question 7 (Functions)

What is the expected output of the following code?

```
num = 1

def func():
    num = num + 3
    print(num)

func()

print(num)
```

A. `4 1`

B. The code is erroneous

C. `4 4`

D. `1 4`

E. `1 1`

Solution 7

In [ ]:
```
"""
num = 1

def func():
    num = num + 3    # 4 <- 1 + 3
    print(num)       # 4
```

```
func()              # 4

print(num)          # 4
"""
# 4 4

# UnboundLocalError:
# local variable 'num' referenced before assignment
```

In [ ]:
```
# Solución
# B
```

Question 8 (Error Handling)

What is the expected behavior of the following program?

```
try:
    print(5/0)
    break
except:
    print("Sorry, something went wrong...")
except (ValueError, ZeroDivisionError):
    print("Too bad...")
```

A. The program will raise an exception handled by the first except block

B. The program will cause a `ValueError` exception and output the following message: `"Too bad.."`

C. The program will cause a `ValueError` exception and output a default error message

D. The program will cause a `ZeroDivisionError` exception and output a default error message

E. The program will cause a `SyntaxError` exception

Solution 8

In [ ]:
```
"""
try:
    print(5/0)
    break
except:
    print("Sorry, something went wrong...")
except (ValueError, ZeroDivisionError):
    print("Too bad...")
"""
# SyntaxError: 'break' outside loop
```

In [ ]:
```
# Solución
# E
```

Question 9 (data types)

The `0o` prefix means that the number after it is denoted as:

A. binary

B. decimal

C. hexadecimal

D. octal

Solution 9

In [ ]:
```
# Solución
# D
```

Question 10 (control flow)

How many stars will the following code print to the monitor?

```
i = 0
while i <= 3:
    i += 2
    print('*')
```

A. zero

B. three

C. one

D. two

In [ ]:
```
i = 0
while i <= 3:    # ------------------>  0-1-2-3
    i += 2       # i = i + 2  ------->  0-2
    print('*')   # ejecuta 2 veces -->  **
```

In [ ]:
```
# Solución
# D
```

Question 11 (Data Aggregates)

Insert the correct snippet so that the program produces the expected output.

Expected output: `True`

Code:

```
list = [False, True, "2", 3, 4, 5]
#### insert code here
print(b)
```

A. `b = 0 not in list`

B. `b = False`

C. `b = 0 in list`

D. `b = list[0]`

Solution 11

In [1]:
```
# A

list = [False, True, "2", 3, 4, 5]
# #### insert code here
b = 0 not in list
print(b)
```

```
False
```

In [2]:
```
# B

list = [False, True, "2", 3, 4, 5]
# #### insert code here
b = False
print(b)
```

```
False
```

In [3]:
```
# C

list = [False, True, "2", 3, 4, 5]
# #### insert code here
b = 0 in list
print(b)
```

```
True
```

INCISO

In [3]:
```
False + 4
```

Out[3]: 4

In [4]:
```
True + 5
```

Out[4]: 6

In [5]:
```
# explicación

lista = [False, True, "2", 3, 4, 5]
# #### insert code here
for elemento in lista:
    if elemento==0:
        print(elemento)
    else:
        print('el elemento:', elemento, 'no es 0 o False')
```

```
False
el elemento: True no es 0 o False
el elemento: 2 no es 0 o False
el elemento: 3 no es 0 o False
el elemento: 4 no es 0 o False
el elemento: 5 no es 0 o False
```

In [6]:
```
# explicación

lista = [False, True, "2", 3, 4, 5]
# #### insert code here
for elemento in lista:
    if elemento==False:
        print(elemento)
    else:
        print('el elemento:', elemento, 'no es False')
```

```
False
el elemento: True no es False
el elemento: 2 no es False
el elemento: 3 no es False
el elemento: 4 no es False
el elemento: 5 no es False
```

In [7]:
```
# D

list = [False, True, "2", 3, 4, 5]
# #### insert code here
b = list[0]
print(b)
```

```
False
```

In [ ]:
```
# Solución
# C
```

Question 12 (control flow)

What is the expected output of the following code?

```
def func(x):
    return 1 if x % 2 != 0 else 2

print(func(func(1)))
```

A. `2`

B. The code is erroneous

C. `1`

D. `None`

Solution 12

In [9]:
```
def func(x):
    return 1 if x % 2 != 0 else 2

print(func(func(1)))
```

```
1
```

In [10]:
```
# Solución
# C
```

In [11]:
```
1 % 2
```

Out[11]:
```
1
```

In [12]:
```
def func(x):
    if x % 2 != 0:
        return 1
    else:
        return 2
```

In [13]:
```
func(1)
```

Out[13]:
```
1
```

In [14]:
```
# print(func(RESULTADO DEL ANTERIOR func(1)))
print(func(func(1)))
```

```
1
```

Question 13 (Data Aggregates)

What is the expected output of the following code?

```
data = ['Peter', 404, 3.03, 'Wellert', 33.3]
print(data[1:3])
```

A. `[404, 3.03]`

B. None of the above

C. `['Peter', 404, 3.03, 'Wellert', 33.3]`

D. `['Peter', 'Wellert']`

Solution 13

In [15]:
```
data = ['Peter', 404, 3.03, 'Wellert', 33.3]
print(data[1:3])  # 404, 3.03
```

```
[404, 3.03]
```

In [ ]:
```
# Solución
# A
```

Question 14 (Operators)

What will be the output of the following code snippet?

```
x = 2
y = 1
x *= y + 1
print(x)
```

A. `4`

B. `None`

C. `2`

D. `3`

E. `1`

Solution 14

In [ ]:
```
# x += 1 compatible con x = x + 1
```

In [ ]:
```
x = 2
y = 1
x *= y + 1  # x  = x * (y + 1)
print(x)    # 4 <= 2 * (1 + 1)
```

In [ ]:
```
# Solución
# A
```

Question 15 (Control Flow)

What is the expected output of the following code?

```
def func(text, num):
    while num > 0:
        print(text)
    num = num - 1

func('Hello', 3)
```

A.

```
Hello
Hello
Hello
Hello
```

B. An infinite loop

C.

```
Hello
Hello
Hello
```

D.

```
Hello
Hello
```

Solution 15

In [1]:
```
"""def func(text, num):
    while num > 0:
        print(text)
    num = num - 1

func('Hello', 3)"""

# bucle infinito
```

Out[1]:
```
"def func(text, num):\n    while num > 0:\n        print(text)\n    num
= num - 1\n\nfunc('Hello', 3)"
```

In [ ]:
```
# Solución
# B
```

SOLUCION QUE SI FUNCIONARIA

OJO CON LA INDENTACIÓN DENTRO DEL BUCLE WHILE

In [16]:
```
def func(text, num):
    while num > 0:
        print(text)
        num = num - 1

func('Hello', 3)
```

```
Hello
Hello
Hello
```

Question 16 (Control Flow)

What is the expected output of the following code?

```
x = True
y = False
z = False

if not x or y:
    print(1)
elif not x or not y and z:
    print(2)
elif not x or y or not y and x:
    print(3)
else:
    print(4)
```

A. `1`

B. `3`

C. `2`

D. `4`

Solution 16

In [17]:
```
x = True
y = False
z = False
```

```
if not x or y:    # False or False
    print(1)
elif not x or not y and z: # False or (True and False)
    print(2)
elif not x or y or not y and x: # False or False or (True and True)
    print(3) # esto imprime
else:
    print(4)
```

```
3
```

In [ ]:
```
# Solución
# B
```

Question 17 (Functions)

A function definition starts with the keyword:

A. def

B. function

C. fun

In [18]:
```
def mi_funcion():
    pass
```

In [19]:
```
# Solución
# A
```

Question 18 (Operators)

In [20]:
```
3/3, 3//3
```

Out[20]:
```
(1.0, 1)
```

What is the expected output of the following code?

```
x = 1 / 2 + 3 // 3 + 4 ** 2

print(x)
```

A. `8.5`

B. `8`

C. `17`

D. `17.5`

Solution 18

In [ ]:
```
x = 1 / 2 + 3 // 3 + 4 ** 2

#    0.5  +  1     +  16 # 17.5
print(x)
```

In [ ]:
```
# Solución
# D
```

Question 19 (Basics)

What is the expected output of the following code?

```
x = 1
y = 2
x, y, z = x, x, y
z, y, z = x, y, z
print(x, y, z)
```

A. `1 1 2`

B. `2 1 2`

C. `1 2 2`

D. `1 2 1`

Solution 19

In [ ]:
```
x = 1
y = 2
x, y, z = x, x, y   # x=x=1, y=x=1, z=y=2
```

```
z, y, z = x, y, z   # z=x=1, y=y=1, z=z=2
print(x, y, z)      # 1       1       2
```

In [ ]:
```
# Solución
# A
```

Question 20 (Data Aggregates)

What is the expected output of the following code?

```
list1 = [1, 3]
list2 = list1
list1[0] = 4
print(list2)
```

A. `[1, 3]`

B. `[1, 3, 4]`

C. `[1, 4]`

D. `[4, 3]`

Solution 20

In [21]:
```
list1 = [1, 3]
list2 = list1  # list2 = [1, 3]
list1[0] = 4   # list2 = [4, 3]
print(list2)
```

```
[4, 3]
```

In [22]:
```
print('list1: ', list1)
print('list2: ', list2)
```

```
list1:  [4, 3]
list2:  [4, 3]
```

In [ ]:
```
# Solución
# D
```

Question 21 (Data Aggregates)

What will be the output of the following code snippet?

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[::2])
```

A. `[1, 2]`

B. `[1, 3, 5, 7, 9]`

C. `[8, 9]`

D. `[1, 2, 3]`

Solution 21

In [29]:
```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[::2])
```

```
[1, 3, 5, 7, 9]
```

In [ ]:
```
# Solución
# B
```

Question 22 (Data Aggregates)

What is the output of the following snippet?

```
dictionary = {'one': 'two', 'three': 'one', 'two':
'three'}
v = dictionary['one']

for k in range(len(dictionary)):
    v = dictionary[v]

print(v)
```

A. `('one', 'two', 'three')`

B. `three`

C. `two`

D. `one`

Solution 22

In [24]:
```
dictionary = {'one': 'two',
              'three': 'one',
              'two': 'three'}

v = dictionary['one']            # two

# len(dictionary) = 3
# range(3) --> 0-1-2
```

```
for k in range(len(dictionary)):
    v = dictionary[v]

# 0: v = dictionary["two"]   --> three
# 1: v = dictionary["three"] --> one
# 2: v = dictionary["one"]   --> two

print(v)                        # two
```

```
two
```

In [25]:
```
# Solución
# C
```

Question 23 (Functions)

What is the output of the following snippet?

```
def fun(x, y, z):
    return x + 2 * y + 3 * z

print(fun(0, z=1, y=3))
```

A. `0`

B. `9`

C. `3`

D. The snippet is erroneous

Solution 23

In [10]:
```
def fun(x, y, z):
    return x + 2 * y + 3 * z
    #      0 + 2 * 3 + 3 * 1
    #      0 +  6    +   3
    #            9

# PESE A QUE LOS ARGUMENTOS DE LA LLAMADA ESTAN DESORDENADOS,
# SE LES RECONOCE IGUALMENTE

print(fun(0, z=1, y=3))
```

```
9
```

In [ ]:
```
# Solución
# B
```

Question 24 (Operators)

An operator able to check whether two values are not equal is coded as:

A. `not ==`

B. `=/=`

C. `<>`

D. `!=`

Solution 24

In [30]:
```
5 != 6
```

Out[30]:
```
True
```

In [ ]:
```
# Solución
# D
```

Question 25 (Basics)

Which of the following variable names are illegal?

(Select two answers)

A. `TRUE`

B. `True`

C. `true`

D. `and`

Solution 25

In [ ]:
```
# B y D

# variables que no pueden llamarse asi,
# son palabra reservada
```

In [ ]:
```
TRUE = 23
true = 42
# True = 7  # SyntaxError: cannot assign to True
# and = 7   # SyntaxError: invalid syntax
```

Question 26 (Operators)

What would you insert instead of ???

so that the program checks for even numbers?

```
if ???:
    print('x is an even number')
```

A. `x % 2 == 0`

B. `x % 2 == 1`

C. `x % 1 == 2`

D. `x % x == 0`

E. `x % 'even' == True`

Solution 26

In [31]:
```
# comprobación
for x in range(7):
    if x % 2 == 0:
        print(x, 'is an even number')
```

```
0 is an even number
2 is an even number
4 is an even number
6 is an even number
```

In [32]:
```
# Solución
# A
```

IMPORTANTE

- EVEN => PAR
- ODD => IMPAR

PREVIO AL EJERCICIO 27

EJEMPLO PREVIO CON data[::2]

In [33]:
```
# de un ejemplo previo vemos que seleccionando:
# data[::2]

# lo que tenemos es aquellos elementos desde el primero (índice 0) de 2 e

a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[::2])
```

```
[1, 3, 5, 7, 9]
```

EJEMPLO CON yield

ITERABLES, GENERATORS, YIELD

Iterables

Listas

Cuando creas una lista es posible leer los elementos (items) 1 a 1.

A ello se le llama iteración

In [34]:
```
# ejemplo

listado = [10, 20, 30]
for i in listado:
    print(i)
```

```
10
20
30
```

list comprehension

In [ ]:
```
# list comprehension es el bucle FOR + IF en una sola línea

# en este caso solamente bucle FOR y todo almacenado en una lista
```

In [35]:
```
listado = [i for i in range(3)]
listado
```

Out[35]: [0, 1, 2]

In [36]:
```
for i in listado:
    print(i)
```

```
0
1
2
```

Generators

In [ ]:
```
# similar a list comprehension

# en vez de [] llevan ()

# SOLO se pueden ejecutar 1 vez

# después olvidan lo que tenían almacenado

# entonces son iterables, pero no se almacenan

# se dice que se generan "on the fly"
```

In [37]:
```
"""
listado = [i for i in range(3)]     LIST COMPREHENSION
listado
"""

generador = (i for i in range(3))   # RANGE(3) ==> 0,1,2
generador
```

Out[37]:
```
<generator object <genexpr> at 0x7f1b84114c80>
```

In [38]:
```
type(generador)
```

Out[38]:
```
generator
```

In [39]:
```
for elemento in generador:
    print(elemento)
```

```
0
1
2
```

In [40]:
```
for j in generador:
    print(j)
```

In [ ]:
```
# esta segunda vez no ejecuta
```

Yield

In [ ]:
```
# yield es una keyword
# es usada como por ejemplo return pero en el que la función retornará un
```

In [41]:
```
def create_generator():
    valores = range(3)      # 0-1-2
    for i in valores:       # 0-1-2
        yield i*i           # yield (0-1-4)
```

In [42]:
```
generador_2 = create_generator()   # create a generator
print(generador_2)                 # generador is an object!
```

```
<generator object create_generator at 0x7f1b840b7040>
```

In [43]:
```
for j in generador_2:

```

```
    print(j)
```

```
0
1
4
```

In [44]:
```
for x in generador_2:
    print(x)
```

In [ ]:
```
# o incluso..
```

In [45]:
```
def create_generator():
    valores = range(3)    # 0-1-2
    for i in valores:     # 0-1-2
        yield i*i         # yield (0-1-4)
```

In [46]:
```
for j in create_generator():
    print(j)
```

```
0
1
4
```

Question 27 (Functions)

CONTENIDO EXTRA AL FINAL DEL DOCUMENTO

What is the expected output of the following code?

```
def func(data):
    for d in data[::2]:
        yield d

for x in func('abcdef'):
    print(x, end='')
```

A. `bdf`

B. `abcdef`

C. `ace`

D. An empty line

Solution 27

In [5]:
```
def func(data):
    for d in data[::2]:
        yield d
```

```
for x in func('abcdef'):
    print(x, end='')
```

```
ace
```

In [ ]:
```
# Solución
# C
```

Question 28 (Basics)

Python is an example of:

A. a natural language

B. a high-level programming language

C. a machine language

Solution 28

In [ ]:
```
# Solución
# B
```

Question 29 (Functions)

What is the expected output of the following code?

```
def fun():
    return True
x = fun(False)
print(x)
```

A. `True`

B. `False`

C. `1`

D. The program will cause an error

Solution 29

In [ ]:
```
"""
def fun():
    return True
```

```
x = fun(False)
print(x)
"""

# TypeError: fun() takes 0 positional arguments but 1 was given
```

In [ ]:
```
# Solución
# D
```

Question 30 (Operators)

What value will be assigned to the x variable?

```
z = 3
y = 7
x = y < z and z > y or y > z and z < y
```

A. `1`

B. `0`

C. `False`

D. `True`

Solution 30

In [ ]:
```
z = 3
y = 7
x   =  y < z and z > y  or  y > z and z < y
# x = (False and False) or (True  and True) => True
# x =         F         or       T
# x =                   T
```

In [47]:
```
z = 3
y = 7
x = y < z and z > y or y > z and z < y
print(x)
```

```
True
```

In [ ]:
```
# Solución
# D
```

Gracias por la atención

Isabel Maniega