Contenido creado por: Isabel Maniega

Question 1

Select the true statements about the try-except block in relation to the following example.

```
try:
    # Some code is here...
except:
    # Some code is here...
```

(Select three answers.)

A. If you suspect that a snippet may raise an exception, you should place it in the try block

B. The code that follows the except statement will be executed if the code in the try clause runs into an error

C. The code that follows the try statement will be executed if the code in the except clause runs into an error

D. if there is a syntax error in the code located in the try block, the except branch will not handle it, and a SyntaxError exception will be raised instead

Solution 1

In [2]:
```
# A, B y D
```

Question 2 (Basics)

What is the expected behavior of the following program?

```
print("Hello!")
```

A. The program will generate an error on the screen

B. The program will output `Hello!` to the screen

C. The program will output `('Hello!')` to the screen

D. The program will output `"Hello!"` to the screen

Solution 2

In [4]:
```
print("Hello!")
```

```
Hello!
```

In [5]:
```
# Solución
# B
```

Question 3 (Control Flow)

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

```
data = [[x for x in range(y)] for y in range(3)]

for d in data:
    if len(d) < 2:
        print('*')
```

A. two

B. zero

C. three

D. one

Solution 3

In [7]:
```
data = [[x for x in range(y)] for y in range(3)]
data
```

Out[7]:
```
[[], [0], [0, 1]]
```

In [8]:
```
# que viene de:

# sabemos que range(3) es 0-1-2
# entonces:
# for y in range(3) ==> y=0, y=1, y=2
```

In [9]:
```
import numpy as np
np.arange(0)
```

Out[9]:
```
array([], dtype=int64)
```

In [10]:
```
[x for x in range(0)]    # range(0) está vacío, sin elementos, longitud 0
```

Out[10]: []

In [11]:
```
[x for x in range(1)]
```

Out[11]: [0]

In [12]:
```
[x for x in range(2)]
```

Out[12]: [0, 1]

In [13]:
```
# ahora vemos el bucle for
```

In [14]:
```
data
```

Out[14]: [[], [0], [0, 1]]

In [15]:
```
for d in data:
    if len(d) < 2:
        print('d=', d)
        print('*')
    else:
        print('\n')
        print('d de longitud >=2', d)
```

```
d= []
*
d= [0]
*


d de longitud >=2 [0, 1]
```

In [16]:
```
# todo de un paso..
```

In [17]:
```
data = [[x for x in range(y)] for y in range(3)]

# range(3) -> 0,1,2 => 0, 1, 2
#
print(data)

for d in data:
    if len(d) < 2:
        print('*')
```

```
[[], [0], [0, 1]]
*
*
```

In [18]:
```
# Solución
# A
```

Question 4 (Operators)

What is the expected output of the following code?

```
print(type(1 / 2))
```

A. `<type 'int'>`

B. `<type 'number'>`

C. `<type 'float'>`

D. `<type 'double'>`

E. `<type 'tuple'>`

Solution 4

In [20]:
```
print(type(1 / 2))
```

```
<class 'float'>
```

In [21]:
```
# Solución
# C
```

Question 5 (Functions)

What is the expected output of the following code?

```
def func(x):
    global y
    y = x * x
    return y

func(2)
print(y)
```

A. `4`

B. `2`

C. The code is erroneous

D. `None`

Solution 5

In [23]:
```
def func(x):
    global y
    y = x * x
    return y
```

```
func(2)  # return 4
print(y) # 4
```

```
4
```

In [24]:
```
# Solución
# A
```

Aclaraciones:

si lo ponemos en celdas separadas lo vemos 2 veces

In [25]:
```
def func(x):
    global y
    y = x * x
    return y


func(2)  # return 4
```

Out[25]: 4

In [26]:
```
print(y) # 4
```

```
4
```

Aclaraciones:

si ponemos 2 print, veremos como lo imprime 2 veces en la misma celda

In [27]:
```
def func(x):
    global y
    y = x * x
    return y


print(func(2))  # return 4
print(y) # 4
```

```
4
4
```

Question 6 (Operators)

OJO NO ES ELEVADO A COMO CALCULADORA

ES XOR

What is the expected output of the following code?

```
x = 0
y = 1
x = x ^ y
y = x ^ y
```

```
y = x ^ y
print(x, y)
```

A. `1 0`

B. `0 0`

C. `0 1`

D. `1 1`

E. The code is erroneous

Solution 6

In [29]:
```
# NO CONFUNDIR !!!!!

print('XOR (de 0^1) :', 0^1, )
print('elevado a (de por ejemplo 0**1):', 0**1)
```

```
XOR (de 0^1) : 1
elevado a (de por ejemplo 0**1): 0
```

In [30]:
```
'''
TABLA DE VERDAD DE OR EXCLUSIVA (XOR)

A B  S
0 0  0
0 1  1
1 0  1
1 1  0

Hay 1 en la salida (S) SI y solo SI,
1 de las entradas está a 1,
PERO NO LAS 2 AL MISMO TIEMPO
"es exclusiva de una o de la otra"
'''
```

Out[30]:
```
'\nTABLA DE VERDAD DE OR EXCLUSIVA (XOR)\n\nA B  S\n0 0  0\n0 1  1\n1 0
1\n1 1  0\n\nHay 1 en la salida (S) SI y solo SI,\n1 de las entradas est
á a 1, \nPERO NO LAS 2 AL MISMO TIEMPO\n"es exclusiva de una o de la otr
a"\n'
```

In [31]:
```
print('XOR:', 0^0, 0^1, 1^0, 1^1)
```

```
XOR: 0 1 1 0
```

In [32]:
```
x = 0
y = 1
x = x ^ y    # x = 0 ^ 1 => x = 1
y = x ^ y    # y = 1 ^ 1 => y = 0
y = x ^ y    # y = 1 ^ 0 => y = 1
print(x, y)  # x = 1   y = 1
```

```
1 1
```

In [33]:
```
# Solución
```

```
# D
```

Question 7 (Data Aggregates)

How many elements does the L list contain?

```
L = [i for i in range(-1, -2)]
```

A. zero

B. two

C. three

D. one

Solution 7

In [2]:
```
L = [i for i in range(-1, -2)]

# compruebo
print(L)      # []
print(len(L)) # 0
```

```
[]
0
```

In [36]:
```
# Solución
# A
```

Question 8 (Data Types)

Consider the following python code:

```
x1 = '23'
y1 = 7
z1 = x1 * y1

x2 = 42
y2 = 7
z2 = x2 / y2

x3 = 4.7
y3 = 1
z3 = x3 / y3
```

What are the data types of the variables z1, z2 and z3?

A. `str, float, float`

B. `str, int, int`

C. `str, int, float`

D. `str, str, str`

Solution 8

In [38]:
```
x1 = '23'
y1 = 7
z1 = x1 * y1    # 23 (7 veces) => str

x2 = 42
y2 = 7
z2 = x2 / y2    # 6.0          => float

x3 = 4.7
y3 = 1
z3 = x3 / y3    # 4.7          => float
```

In [39]:
```
# type(z1, z2, z3)

# TypeError:
# type.__new__() argument 2 must be tuple, not float
```

In [40]:
```
z1, z2, z3
```

Out[40]:
```
('23232323232323', 6.0, 4.7)
```

In [41]:
```
type(z1), type(z2), type(z3)
```

Out[41]:
```
(str, float, float)
```

In [42]:
```
# Solución
# A
```

Question 9 (Operators)

The `**` operator ...

A. performs floating-point multiplication

B. performs exponentiation

C. does not exist

D. performs duplicated multiplication

Solution 9

In [44]:
```
2**3  # 2*2*2
```

Out[44]: 8

In [45]:
```
# Solución
# B
```

Question 10 (Data Aggregates)

What is the expected output of the following code?

```
data = (1,) * 3
data[0] = 2
print(data)
```

A. `(2, 2, 2)`

B. `(2, 1, 1)`

C. `(1, 1, 1)`

D. The code is erroneous

Solution 10

In [47]:
```
data = (1,) * 3
data
```

Out[47]: (1, 1, 1)

In [48]:
```
type(data)
```

Out[48]: tuple

In [49]:
```
"""data = (1,) * 3
data[0] = 2
print(data)"""

# TypeError:
# 'tuple' object does not support item assignment

# SIENDO UNA TUPLA =====> NO Podemos modificar valores
```

Out[49]: 'data = (1,) * 3\ndata[0] = 2\nprint(data)'

In [50]:
```
# Solución
# D
```

---

explicación

no es lo mismo el primer ejemplo que el segundo

uno se refiere a 1 tupla, y el otro 1 entero

In [51]:
```
# tupla, de 1 solo elemento, el número 1
```

In [52]:
```
data_tupla = (1,) * 3
data_tupla
```

Out[52]: (1, 1, 1)

In [53]:
```
type(data_tupla)
```

Out[53]: tuple

In [54]:
```
# entero, podríamos quitar el paréntesis de hecho
```

In [55]:
```
data_entero = (1) * 3
data_entero
```

Out[55]: 3

In [56]:
```
type(data_entero)
```

Out[56]: int

In [57]:
```
# tupla, son 2 elementos, con paréntesis
```

In [58]:
```
data_tupla2 = (1,2) * 3
data_tupla2
```

Out[58]: (1, 2, 1, 2, 1, 2)

In [59]:
```
type(data_tupla2)
```

Out[59]: tuple

Otra posible solución al ejercicio...

In [3]:
```
data = (1,) * 3
data = list(data)
data[0] = 2
print(data)

tupla = tuple(data)
print(tupla)
```

```
[2, 1, 1]
(2, 1, 1)
```

Question 11 (Data Aggregates)

What is the output of the following snippet?

```
my_list = [3, 1, -2]
print(my_list[my_list[-1]])
```

A. `1`

B. `3`

C. `-2`

D. `-1`

Solution 11

In [61]:
```
my_list = [3, 1, -2]
print(my_list[my_list[-1]])
#                 [-2]
# my_list[-2] => 1
```

```
1
```

In [62]:
```
# Solución
# A
```

Question 12

What is the expected output of the following code?

```
def func1(a):
    return a ** a

def func2(a):
    return func1(a) * func1(a)

print(func2(2))
```

A. The code is erroneous

B. `16`

C. `2`

D. `4`

Solution 12

In [64]:
```
def func1(a):
    return a ** a

def func2(a):
    return func1(a) * func1(a)

print(func2(2))

# func1(2) * func1(2)
#  2**2    *  2**2
#    4     *   4
#         16
```

16

In [65]:
```
# Solución
# B
```

Question 13 (Data Types)

Which of the following is the output of the below Python code?

```
str = 'Hello World'
print(str[::-1])
```

A. `World`

B. `dlroW olleH`

C. `Hello World`

D. `Hello`

Solution 13

In [67]:
```
str = 'Hello World'
print(str[::-1])
```

```
dlroW olleH
```

In [68]:
```
# Solución
# B
```

Question 14 (Control Flow)

You are creating a Python script to evaluate input

and check for upper and lower case.

Code segment 1:

```
    else:
        print(name, 'is mixed case.')
```

Code segment 2:

```
    else:
        print(name, 'is lower case.')
```

Code segment 3:

```
    name = input('Enter your name: ')
```

Code segment 4:

```
    else:
        print(name, 'is upper case.')
```

Code segment 5:

```
    elif name.upper() == name:
        print(name, 'is all upper case.')
```

Code segment 6:

```
    if name.lower() == name:
        print(name, 'is all lower case.')
```

Which four code segments should you use to develop the solution?

A.

```
Code segment 3
Code segment 6
Code segment 5
Code segment 1
```

B.

```
Code segment 1
Code segment 3
Code segment 5
Code segment 6
```

C.

```
Code segment 3
Code segment 6
Code segment 5
Code segment 4
```

D.

```
Code segment 3
Code segment 6
Code segment 5
Code segment 2
```

Solution 14

In [70]:
```python
# Code segment 3:
name = input('Enter your name: ')

# Code segment 6:
if name.lower() == name:
    print(name, 'is all lower case.')

# Code segment 5:
elif name.upper() == name:
    print(name, 'is all upper case.')

# Code segment 1:
else:
    print(name, 'is mixed case.')

# 3 6 5 1
```

```
Enter your name: Jose
Jose is mixed case.
```

In [71]:
```python
# Solución
```

```
# A
```

Question 15 (Data Aggregates)

What is the expected output of the following code?

```
data = {'z': 23, 'x': 7, 'y': 42}

for _ in sorted(data):
    print(data[_], end=' ')
```

A. `7 23 42`

B. `7 42 23`

C. `42 23 7`

Solution 15

In [73]:
```
# sorted data

data = {'z': 23, 'x': 7, 'y': 42}
data
```

Out[73]:
```
{'z': 23, 'x': 7, 'y': 42}
```

In [74]:
```
data = {'z': 23, 'x': 7, 'y': 42}
sorted(data)

# son las claves en orden alfabético
```

Out[74]:
```
['x', 'y', 'z']
```

In [75]:
```
# el propio ejercicio
```

In [76]:
```
data = {'z': 23, 'x': 7, 'y': 42}         # diccionario

for _ in sorted(data):                    # ordena las claves en orden al
    # x y z
    print(data[_], end=' ')               # data['clave'] => devuelve el
                                          # data['key1'] ==> value1 (ejem
    # 7 42 23                             # con un espacio entre valores
```

```
7 42 23
```

In [77]:
```
# Solución
# B
```

algunos ejemplos similares...

ejemplo similar 1: compruebo cómo funciona sorted(data)

In [78]:
```
data = {'z': 23, 'x': 7, 'y': 42}

for clave in sorted(data):
    print(clave)
```

```
x
y
z
```

ejemplo similar 2: imprimo los valores asociados a cada clave

In [79]:
```
data = {'z': 23, 'x': 7, 'y': 42}

for clave in sorted(data):
    print(data[clave])
```

```
7
42
23
```

ejemplo similar 3: uso data.keys() y data.values()

In [80]:
```
data = {'z': 23, 'x': 7, 'y': 42}
print('data.keys(): ', data.keys())
print('data.values(): ', data.values())
```

```
data.keys():  dict_keys(['z', 'x', 'y'])
data.values():  dict_values([23, 7, 42])
```

ejemplo similar 4: uso data.keys() para obtener los valores

In [81]:
```
data = {'z': 23, 'x': 7, 'y': 42}

for key in data.keys():
    print(data[key])
```

```
23
7
42
```

Question 16 (Data Aggregates)

What is the expected output of the following code?

```
box = {}
jars = {}
crates = {}

box['biscuit'] = 1
box['cake'] = 3

jars['jam'] = 4
```

```
crates['box'] = box
crates['jars'] = jars

print(len(crates[box]))
```

A. `1`

B. `3`

C. `4`

D. The code is erroneous

E. `2`

Solution 16

In [83]:
```
box = {}
jars = {}
crates = {}

box['biscuit'] = 1      # box = {'biscuit': 1}
box['cake'] = 3         # box = {'biscuit': 1, 'cake': 3}

jars['jam'] = 4         # jars = {'jam': 4}

crates['box'] = box     # crates = {'box': box}
crates['jars'] = jars   # crates = {'box': box, 'jars': jars}

# print(len(crates[box]))     # no funciona
```

In [84]:
```
# Solución
# D
```

explicación

In [85]:
```
# Otro ejemplo:


box = {}
jars = {}
crates = {}

print('box inicial:', box)
box['biscuit'] = 1      # box = {'biscuit': 1}
box['cake'] = 3         # box = {'biscuit': 1, 'cake': 3}
print('box final:', box)
print('\n')
# ------------------------------------------------------------
print('jars inicial:', jars)
jars['jam'] = 4         # jars = {'jam': 4}
print('jars final:', jars)
print('\n')
# ------------------------------------------------------------
print('crates incicial:', crates)
```

```
crates['box'] = box     # crates = {'box': box}
crates['jars'] = jars   # crates = {'box': box, 'jars': jars}
print('crates final'    , crates)
```

```
box inicial: {}
box final: {'biscuit': 1, 'cake': 3}


jars inicial: {}
jars final: {'jam': 4}


crates incicial: {}
crates final {'box': {'biscuit': 1, 'cake': 3}, 'jars': {'jam': 4}}
```

In [86]:
```
print(crates['box'])
```

```
{'biscuit': 1, 'cake': 3}
```

In [87]:
```
print(len(crates['box']))
```

```
2
```

In [3]:
```
# print(len(crates[box]))        # no funciona, falta la comilla de la ke
```

Question 17 (Control Flow)

What is the expected output of the following code?

```
data = [[42, 17, 23, 13], [11, 9, 3, 7]]
res = data[0][0]
for da in data:
    for d in da:
        if res > d:
            res = d

print(res)
```

A. `42`

B. The code is erroneous

C. `3`

D. `13`

Solution 17

In [90]:
```
#----obtengo res-----
```

In [91]:
```
data = [[42, 17, 23, 13],
        [11, 9, 3, 7]]
```

```
res = data[0][0]     # res = 42    (fila 0 y columna 0 de la matriz de 2
res
```

Out[91]: 42

In [92]:
```
# ----observo da en data-------
```

In [93]:
```
for da in data:
    print(da)
```

```
[42, 17, 23, 13]
[11, 9, 3, 7]
```

In [94]:
```
len(data)
```

Out[94]: 2

In [95]:
```
# -----observo d en da---------
```

In [96]:
```
for da in data:            # [[42, 17, 23, 13], [11, 9, 3, 7]]
    for d in da:           # todos los números por separado
        print(d)
```

```
42
17
23
13
11
9
3
7
```

In [97]:
```
# ---- el propio ejercicio completo -------
```

In [98]:
```
data = [[42, 17, 23, 13],
        [11, 9, 3, 7]]

res = data[0][0]           # res = 42
for da in data:            # [[42, 17, 23, 13], [11, 9, 3, 7]]
    for d in da:           # para cada numero de la lista
        if res > d:        # si d < res:  (si ese número es más pequeño q
            res = d        # el nuevo mínimo de la lista es ese número
print(res)
```

```
3
```

In [99]:
```
# Solución
# C
```

Question 18 (Operators)

What is the expected output of the following code?

```
list1 = ['Peter', 'Paul', 'Mary', 'Jane']
list2 = ['Peter', 'Paul', 'Mary', 'Jane']

print(list1 is not list2)
print(list1 != list2)

list1 = list2

print(list1 is not list2)
print(list1 != list2)
```

A.

```
True
False
False
False
```

B.

```
True
False
True
False
```

C.

```
True
True
False
False
```

D.

```
True
False
False
True
```

Solution 18

In [101…

```
list1 = ['Peter', 'Paul', 'Mary', 'Jane']
list2 = ['Peter', 'Paul', 'Mary', 'Jane']

print(list1 is not list2)     # True
print(list1 != list2)         # False

list1 = list2

print(list1 is not list2)     # False
print(list1 != list2)         # False
```

```
True
False
False
False
```

In [102…
```
# Solución
# A
```

ejemplo similar 1: cambiando alguna cosilla

In [103…
```
list1 = ['Peter', 'Paul', 'Mary', 'Jane']
list2 = ['Peter', 'Paul', 'Mary', 'Jane']
print('list1 inicial:', list1)
print('list2 inicial:', list2)

print(list1 is list2)         # False
print(list1 == list2)         # True

# --------ahora hacemos: list1 = list2 ----------------

list1 = list2
print('LIST_1 cuando haces list1 = list2:', list1)
print('LIST_2 cuando haces list1 = list2:', list2)

print(list1 is list2)         # True
print(list1 == list2)         # True
```

```
list1 inicial: ['Peter', 'Paul', 'Mary', 'Jane']
list2 inicial: ['Peter', 'Paul', 'Mary', 'Jane']
False
True
LIST_1 cuando haces list1 = list2: ['Peter', 'Paul', 'Mary', 'Jane']
LIST_2 cuando haces list1 = list2: ['Peter', 'Paul', 'Mary', 'Jane']
True
True
```

ejemplo similar 2: cambiando alguna cosilla, y apendizando algún valor

In [104…
```
list1 = ['Peter', 'Paul', 'Mary', 'Jane']
list2 = ['Peter', 'Paul', 'Mary', 'Jane']
print('list1 inicial:', list1)
print('list2 inicial:', list2)

print(list1 is list2)         # False
print(list1 == list2)         # True
print('\n')
# --------ahora hacemos: list1 = list2 ----------------

list1 = list2
print('LIST_1 cuando haces list1 = list2:', list1)
print('LIST_2 cuando haces list1 = list2:', list2)
print('\n')

# --------apendizamos algun valor:---------------------

list1.append('Jose')

print('LIST_1 cuando apendizas en list1:', list1)
print('LIST_2 cuando apendizas en list1:', list2)
```

```
print('\n')

# y ahora...
print(list1 is list2)         # True
print(list1 == list2)         # True
```

```
list1 inicial: ['Peter', 'Paul', 'Mary', 'Jane']
list2 inicial: ['Peter', 'Paul', 'Mary', 'Jane']
False
True


LIST_1 cuando haces list1 = list2: ['Peter', 'Paul', 'Mary', 'Jane']
LIST_2 cuando haces list1 = list2: ['Peter', 'Paul', 'Mary', 'Jane']


LIST_1 cuando apendizas en list1: ['Peter', 'Paul', 'Mary', 'Jane', 'Jos
e']
LIST_2 cuando apendizas en list1: ['Peter', 'Paul', 'Mary', 'Jane', 'Jos
e']


True
True
```

Question 19 (Functions)

Which of the literals below is not a valid function name?

A. `Func_1_tion`

B. `Function1`

C. `_function1`

D. `1function`

E. `Function_1`

Solution 19

In [106…
```
# D
```

In [107…
```
"""
def 1funcion():
    pass
"""
# SyntaxError: invalid syntax

# COMIENZA POR UN NÚMERO, No está permitido
```

Out[107… `'\ndef 1funcion():\n    pass\n'`

Question 20 (Functions)

What is the expected output of the following code?

```
def get_names():
    names = ['Peter', 'Paul', 'Mary', 'Jane', 'Steve']
    return names[2:]

def update_names(names):
    res = []
    for name in names:
        res.append(name[:3].upper())
    return res

print(update_names(get_names()))
```

A. `['JAN', 'STE']`

B. `['JA', 'ST']`

C. `['MAR', 'JAN', 'STE']`

D. `['MA', 'JA', 'ST']`

Solution 20

In [109…
```
# concepto previo 1
names = ['Peter', 'Paul', 'Mary', 'Jane', 'Steve']
print(names[2:])   # ['Mary', 'Jane', 'Steve']
```

```
['Mary', 'Jane', 'Steve']
```

In [110…
```
# concepto previo 2
nombre = 'Peter'
nombre[:3]  # 'Pet'
```

Out[110…
```
'Pet'
```

In [111…
```
# el propio ejercicio
```

In [112…
```
def get_names():
    names = ['Peter', 'Paul', 'Mary', 'Jane', 'Steve']
    return names[2:]   # ['Mary', 'Jane', 'Steve']

def update_names(names):
    res = []
    for name in names:
        res.append(name[:3].upper())  # ['MAR', 'JAN', 'STE']
    return res
```

```
print(update_names(get_names()))
```

```
['MAR', 'JAN', 'STE']
```

In [113…
```
# Solución
# C
```

Question 21 (Operators)

What is the expected output of the following code?

```
a = 1
b = 0
c = a & b
d = a | b
e = a ^ b
print(c + d + e)
```

A. `1`

B. `2`

C. `0`

D. `3`

Solution 21

In [115…
```
# https://realpython.com/python-bitwise-operators/
"""

Operator        Example Meaning
&       a & b   Bitwise AND
|       a | b   Bitwise OR
^       a ^ b   Bitwise XOR (exclusive OR)
~       ~a      Bitwise NOT
<<      a << n  Bitwise left shift
>>      a >> n  Bitwise right shift
"""
```

Out[115…
```
'\n\nOperator\tExample\tMeaning\n&\ta & b\tBitwise AND\n|\ta | b\tBitwis
e OR\n^\ta ^ b\tBitwise XOR (exclusive OR)\n~\t~a\tBitwise NOT\n<<\ta <<
n\tBitwise left shift\n>>\ta >> n\tBitwise right shift\n'
```

In [116…
```
a = 1                       # => a = 1
b = 0                       # => b = 0
c = a & b          # 1 AND 0  => c = 0
d = a | b          # 1 OR  0  => d = 1
e = a ^ b          # 1 XOR 0  => e = 1

print(c + d + e)       # 0 + 1 + 1 = 2
```

2

In [117…
```
# Solución
# B
```

Question 22 (Functions)

What is the expected output of the following code?

```
def func1(param):
    return param

def func2(param):
    return param * 2

def func3(param):
    return param + 3

print(func1(func2(func3(1))))
```

A. `8`

B. `1`

C. `6`

D. `3`

Solution 22

In [119…
```
def func1(param):
    return param

def func2(param):
    return param * 2

def func3(param):
    return param + 3

print(func1(func2(func3(1))))
#     func1(func2(1+3)     # retorna parámetro + 3
#     func1(func2(4))      # entonces..
#     func1(2*4)           # retorna parámetro * 2
#     func1(8)             # entonces..
#          8               # retorna el propio parámetro
```

8

In [120…
```
# Solución
# A
```

Question 23 (Operators)

Consider the following code.

```
x = float('23.42')
```

Which of the following expressions will evaluate to `2`?

A. `bool(x)`

B. `str(x)`

C. `bool(x) + True`

D. `int(x) + False`

Solution 23

In [122…
```
x = float('23.42')     # float de un string es float
x
```

Out[122…
```
23.42
```

In [123…
```
# A
bool(x)                # si hay elementos, da 1
```

Out[123…
```
True
```

In [124…
```
# B
# str(x)

# TypeError: 'str' object is not callable
```

In [125…
```
# C
bool(x) + True
```

Out[125…
```
2
```

In [126…
```
int(x)
```

Out[126…
```
23
```

In [127…
```
False
```

Out[127…
```
False
```

In [128…
```
0 + False
```

Out[128…
```
0
```

In [129…
```
# D
int(x) + False
```

Out[129… 23

In [130…
```
# Solución
# C
```

ejemplos para la explicación

In [131…
```
k1 = ''
print('bool(k1):', bool(k1))     # no hay nada

k2 = ' '
print('bool(k2)', bool(k2))      # hay un espacio

k3 = 13
print('bool(k3)', bool(k3))      # un número (positivo)

k4 = -15
print('bool(k4)', bool(k4))      # un número (negativo)
```

```
bool(k1): False
bool(k2) True
bool(k3) True
bool(k4) True
```

In [132…
```
True + 1    # True = 1
```

Out[132… 2

In [133…
```
True + 3
```

Out[133… 4

In [134…
```
False + 1   # False = 0
```

Out[134… 1

In [135…
```
False - 2
```

Out[135… -2

Question 24 (Data Types)

What is the expected output of the following code?

```
print(chr(ord('p') + 3))
```

A. `s`

B. `q`

C. `t`

D. `r`

Solution 24

In [137…
```
ord('p')
```

Out[137… 112

In [138…
```
print(chr(ord('p') + 3))  # s
```

```
s
```

In [139…
```
# Solución
# A
```

Question 25 (Data Types)

You want the name, the user puts in to be written back to the monitor.

What snippet would you insert in the line indicated below:

```
print('Enter Your Name: ')
# insert your code here
print(name)
```

A. `name = input`

B. `input('name')`

C. `input(name)`

D. `name = input()`

Solution 25

In [141…
```
print('Enter Your Name: ')
name = input()
print(name)
```

```
Enter Your Name:
Jose
Jose
```

In [142…
```
# Solución
# D
```

Question 26 (Control Flow)

What would you insert instead of ???

so that the program prints TRUE to the monitor?

```
w = 7
x = 3
y = 4
z = True
a = w + x * y
b = w + x / z

if ???:
    print('TRUE')
else:
    print('FALSE')
```

A. `a < b`

B. `a <= b`

C. `a > b`

D. `a == b`

Solution 26

In [144…
```
True + 1  # True en las sumas => 1
```

Out[144… 2

In [1]:
```
w = 7
x = 3
y = 4
z = True           # 1 en las sumas
a = w + x * y      # 7 + 3 * 4 => a = 14
b = w + x / z      # 7 + 3 / 1 => b = 10

if a > b:
    print('TRUE')
else:
    print('FALSE')
```

TRUE

In [146…
```
# Solución
# C
```

Question 27 (Error Handling)

Which of the following snippets shows the correct way

of handling multiple exceptions in a single except clause?

A.

```
except: TypeError, ValueError, ZeroDivisionError
    some code.
```

B.

```
except: (TypeError, ValueError, ZeroDivisionError)
    some code.
```

C.

```
except TypeError, ValueError, ZeroDivisionError
    some code.
```

D.

```
except TypeError, ValueError, ZeroDivisionError:
    some code.
```

E.

```
except (TypeError, ValueError, ZeroDivisionError)
    some code.
```

F.

```
except (TypeError, ValueError, ZeroDivisionError):
    some code.
```

Solution 27

In [148…
```
# https://www.tutorialspoint.com/How-to-use-the-except-clause-with-multip

# Except(Exception1, Exception2,…ExceptionN) as e:
```

In [149…
```
# Solución
# F
```

Question 28 (Data Aggregates)

What is the expected output of the following code?

```
data = {}
data[1] = 1
data['1'] = 2
data[1.0] = 4

res = 0
for d in data:
    res += data[d]

print(res)
```

A. `6`

B. `7`

C. `3`

D. The code is erroneous

Solution 28

In [151…
```
# -----primero analizo el diccionario-----
```

In [152…
```
data = {}
print('data:', data)
print('\n')

data[1] = 1
print('data:', data)
print('\n')

data['1'] = 2
print('data:', data)
print('\n')

data[1.0] = 4                # aunque venga en formato decimal, reconoce
print('data:', data)         # y modifica por tanto, el valor asociado a
print('\n')
```

```
data: {}


data: {1: 1}


data: {1: 1, '1': 2}


data: {1: 4, '1': 2}
```

In [153…
```
# bucle for...cambiando las claves..
```

In [154…
```
# imprimo las claves

data = {1:4, '2':2}
for d in data:
    print(d)
```

```
1
2
```

In [155…
```
# imprimo los valores

data = {1:4, '2':2}
for d in data:
    print(data[d])
```

```
4
2
```

In [156…
```
# --------el ejercicio completo ----------
```

In [157…
```
data = {}
print(data)

data[1] = 1         # data = {1:1}
print(data)

data['1'] = 2       # data = {1:1, '1': 2}
print(data)

data[1.0] = 4       # data = {1:4, '1':2}
# (reconoce aun siendo decimal)
print(data)

print("\n")

res = 0
for d in data:
    res += data[d]   # res = res + data[d]  ======> va a sumar los 2 valu
    print('d:', d, 'data[d]:', data[d])
    print('res:', res)

print("\n")
print(res)       # ==> dado que suma los values asociados a las keys ==>
```

```
{}
{1: 1}
{1: 1, '1': 2}
{1: 4, '1': 2}


d: 1 data[d]: 4
res: 4
d: 1 data[d]: 2
res: 6


6
```

In [158…
```
# Solución
# A
```

Question 29

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

```
   While True:
       print("1")
```

A. The program will fall into infinite loop, printing `1` on each line

B. The program will print `1`, on one line only

C. The program will run and cause an error

D. The program will not run due to `syntax error`

Solution 29

In [160…
```
"""While True:
    print("1")"""

# la w es mayúscula
```

Out[160… `'While True:\n    print("1")'`

In [161…
```
# Ojo!!!! la w es mayúscula
```

In [162…
```
# Otro ejemplo..

"""while True:
    print("1")"""

# bucle infinito
```

Out[162… `'while True:\n    print("1")'`

In [163…
```
# Solución
# D
```

conclusion/reflexión

hay que leer muy bien las respuestas, y el propio código

pudiéramos pensar que es un bucle infinito por el while True

pero debemos saber que WHILE o While no son palabras reconocidas

en Python la única opción es while con minúsculas todas sus letras

---

Question 30 (Functions)

The following snippet:

```
def func(a, b):
    return a ** a

print(func(2))
```

A. will return `None`

B. will output `2`

C. will output `4`

D. is erroneous

Solution 30

In [165…
```
"""
def func(a, b):
    return a ** a

print(func(2))
"""


# TypeError: func() missing 1 required positional argument: 'b'
```

Out[165… `'\ndef func(a, b):\n    return a ** a\n \nprint(func(2))\n'`

In [166…
```
# Solución
# D
```

conclusión

no importa cuantas variables uses realmente en la función (en la definición)

si pasas 2 variables como argumento, debe saberse el valor de ambas (en la llamada)

Gracias por la atención

Isabel Maniega