Week-2, Graded Assignment (theory)

Problem 1

Question

What will be the output of `print(a)` after executing the below code?

```python
a, b, c, d = input()
```

Input

```
1234
```

Answer

`1`

Solution

Variable	Value
`a`	`'1'`
`b`	`'2'`
`c`	`'3'`
`d`	`'4'`

The statement accepts a string of length `4` and assigns each character to variables `a`, `b`, `c` and `d` in the order it is entered. If the number of characters in the entered string is not equal to `4` , `ValueError` is thrown by the interpreter.

Problem 2

Question

What will be the output of `print(b)` after executing the below code?

```
1234
```

Answer

`2`

Solution

See problem 1

Problem 3

Question

What will be the output of `print(c)` after executing the below code?

```
1234
```

Answer

`3`

Solution

See problem 1

Problem 4

Question

What will be the output of `print(d)` after executing the below code?

```
1234
```

Answer

`4`

Solution

See problem 1

Problem 5

Question

What is the value stored in the variable `x` after the following line of code is executed? Assume that the user gives `False` as the input.

```
x = bool(False)
```

(a) `False`
(b) `True`
(c) `ValueError`
(d) `'False'`

Answer

(a) `False`

Solution

`bool()` is a built-in function that converts a value passed as an argument to a `bool` literal `True` or `False`. If the argument is an empty string, it returns `False` , else it will return `True`. If the value inside the function is a `bool` literal, the same is returned by the function. If in case the argument is an `int` or `float`, `False` is returned when it is `0` or `0.0`, otherwise `True` is returned.

Problem 6

Question

What is the value stored in the variable `x` after the following line of code is executed? Assume that the user gives `False` as the input.

```
x = bool(input())
```

(a) `False`
(b) `True`
(c) `ValueError`
(d) `'False'`

Answer

(b) `True`

Solution

As you know, the function `input()` reads the value passed by the user as a string. Here, the value entered by user is `False`. The `input()`function returns it as a string literal `'False'`. It is then transformed by the function `bool()` to the boolean literal `True` and is stored in the variable `x`.

The given code can be interpreted as:

```
x = bool('False')
# input to bool is a string
# as input is not the empty string, True is the output of bool('False')
```

Below table shows how values are transformed by `bool()` function:

Type	Values	Boolean Conversion
`int`	`0`	`False`
`int`	`1`,`2`,`3`,..., `-1`, `-2`,..., , `10**2`	`True`
`float`	`0.0`, `-0.0`	`False`
`float`	`0.00001`, `2.718`	`True`
`str`	`""`	`False`
`str`	`" "`, `"a"`, `"hello"`	`True`

Use the below code to answer Question number 7,8 and 9.

Code-1

```python
a = int(input())
b = int(input())

if a > 0:
    if b < 0:
        print('OK')
```

Code-2

```python
if int(input()) > 0 and int(input()) < 0:
    print('Check OK')
```

Problem 7

Question

Choose the correct statements. It is a Multiple Select Question (MSQ).

(a) Code-1: Always accepts two inputs

(b) Code-2: Always accepts two inputs

(c) Code-1: If the first input is negative then program completes without printing anything

(d) Code-2: If the first input is negative then program completes without printing anything

(e) Code-1: It is possible to change the value of a by introducing a new line of code before accepting the value of b

Answer

(a) Code-1: Always accepts two inputs

(c) Code-1: If the first input is negative then program completes without printing anything

(d) Code-2: If the first input is negative then program completes without printing anything

(e) Code-1: It is possible to change the value of a by introducing a new line of code before accepting the value of b

Solution

Option (a) is correct. The interpreter executes statements sequentially in a program. In Code-1 block, line-1 accepts user input and store in a variable `a`. On line-2, it accepts another input from console and stores in variable `b`. Therefore, it always accepts two inputs.

Option (c) is correct. In Code-1, if the first input number `a` is negative, the condition `a > 0` evaluates to `False`, the control doesn't enter inside the body of this `if` block and therefore subsequent statement `if b < 0:` is skipped.

Option (d) is correct. If the number entered for the first `input()` call is less than `0`, the condition `int(input()) > 0 and int(input()) < 0` evaluates to `False`. Thus, neither second `input()` is executed, nor the `print('OK')` statements inside the body of this `if` block.

Option (e) is correct. One can change the value of `a` before accepting the second variable `b` as input. Here is an example to reassign `1` to the variable `a` on line-2.

```
1  a = int(input())
2  a = 1
3  b = int(input())
4  if a > 0:
5      if b < 0:
6          print('OK')
```

Problem 8

Question

How many comparisons are done for the following input in Code-1?

```
-1
1
```

Answer

`1`

Solution

In Code-1, only `1` comparison occurs for given input. Since the first input number `a` is `-1`, the condition `a > 0` evaluates to `False`, the control doesn't enter inside the body of this `if` block and therefore subsequent conditional statement is skipped.

Problem 9

Question

How many comparisons are done for the following input in Code-2?

```
1
1
```

Answer

`2`

Solution

In Code-2, `2` comparisons occur. Let us look at the expression in the conditional statement:

`int(input()) > 0 and int(input()) < 0`

Since the first input number is greater than `0`, the first part of the expression `int(input()) > 0` evaluates to `True`. Therefore, the Python interpreter will go ahead and do the second comparison `int(input()) < 0`. This is because the operator is `and` and the operand to the left of this operator evaluates to `True`. In such a case, the operand to the right should also be evaluated to find the value of the entire expression. Refer short-circuit evaluation in Python for more details.

Problem 10

Question

Code-1

```python
if a:
    if b:
        print('OK')
    if c:
        print('OK')
```

Code-2

```python
if a and b or c:
    print('OK')
```

Choose the correct statements, given that a, b and c are boolean values. It is a Multiple Select Question (MSQ).

(a) Code-1: OK will be printed once if either of a or b is True given that c is False

(b) Code-2: OK will be printed once if either of a or b is True given that c is False

(c) Code-1: OK will be printed twice only if all a , b and c are True

(d) Code-2: OK will be printed only if all a , b and c are True

(e) Code-2: OK will be printed if c is True

(f) Code-1 and Code-2 will give same result for same values of a , b and c

Answer

(c) Code-1: OK will be printed twice only if all a , b and c are True

(e) Code-2: OK will be printed if c is True

Solution

Option (c) is correct. In Code-1, the interpreter executes the body of outer `if` block, only if the boolean variable `a` is `True`. The first inner `print('OK')` statement is executed if boolean variable `b` is `True`. Similarly, the second `print('OK')` is executed when the boolean variable `c` is `True`.

Option (e) is correct. In Code-2, the conditional expression `a and b or c` contains two logical operators `and` and `or`. It is parenthesized as `(a and b) or c` according to the precedence rules (`and > or`). This condition evaluates to `True`, if `c` is `True`, irrespective of the values of `a` and `b`.

Use the below code to answer Question number 11,12,13 and 14.

```python
x = int(input())
y = int(input())
z = int(input())

# Block-1 Start
if x > 0 or y > 0 or z > 0:
    if (x > 0 and y > 0) or (y > 0 and z > 0) or (z > 0 and x > 0):
        if x > 0 and y > 0 and z > 0:
            print('P3')
        else:
            print('P2')
    else:
        print('P1')
# Block-1 End

# Block-2 Start
if x < 0 or y < 0 or z < 0:
    if (x < 0 and y < 0) or (y < 0 and z < 0) or (z < 0 and x < 0):
        if x < 0 and y < 0 and z < 0:
            print('N3')
        else:
            print('N2')
    else:
        print('N1')
# Block-2 End
```

Problem 11

Question

What will be the value of X in the output for the given input? It is a Numerical Answer Type (NAT) question.

Input

```
-1
4
1
```

Output

```
PX
NY
```

Answer

`2`

Solution

In the first nested `if` structure (Block-1), the outermost `if` condition checks whether at least one of the variables `x`, `y`, `z` is positive. The intermediate `if` block is executed when two of these variables are positive, otherwise `else` block is executed and `P1` is printed.

Inside the intermediate `if` body, the innermost `if` block is executed, and `P3` is printed when all three variables are positive, otherwise `else` block is executed and `P2` is printed on the console.

Similar analogy for Block-2.

Here is a table showing the input and output relation.

Input	Output	X	Y
One of the numbers is positive	`P1`	1	
Two numbers are positive	`P2`	2	
All three numbers are positive	`P3`	3	
One of the numbers is negative	`N1`		1
Two numbers are negative	`N2`		2
All three numbers are negative	`N3`		3
All three numbers are zero			

Problem 12

Question

What will be the value of Y in the output for the given input? It is a Numerical Answer Type (NAT) question.

Input

```
-1
4
1
```

Output

```
PX
NY
```

Answer

`1`

Solution

See problem 11

Problem 13

Question

When does the above code print no value?

(a) When any two among x , y and z are equal

(b) When all the values of x , y and z are equal

(c) When any one among x , y and z is zero

(d) When all the values of x , y and z are zeros

Answer

(d) When all the values of x , y and z are zeros

Solution

Option (d) is correct. When all variables set to `0`, Outermost `if` block evaluates `False` in both Block-1 and Block-2. Hence, nothing is printed.

Problem 14

Question

For any input `x`, `y` and `z` at least one `else` statement will be executed.

(a) True
(b) False

Answer

(b) False

Solution

Option (b) is correct. In below situations, none of the `else` statement are executed.

- All variables are set to `0` ,
- All variables are positive,
- All variables are negative

Problem 15

Question

```python
import math as ma
```

Select the correct way of accessing the sqrt function from math library.

(a) `math.sqrt()`

(b) `ma.sqrt()`

(c) `sqrt()`

(d) `math.ma.sqrt()`

Answer

(b) `ma.sqrt()`

Solution

A library can be included in the program using the keyword `import`. The `as` keyword allows us to call various functions of a library using a custom alias (name). Here `math` library is imported under the alias `ma`. Hence, (b) `ma.sqrt()` is the correct answer.

Problem 16

Question

What should be input to the following code to get the below output. This is a MCQ type question.

```python
dept    = input()
course  = input()
prefix  = input()
name    = input()
roll_no = input()
name = prefix + " " + name
lib_id = dept[0] + course[0] + roll_no
print("Student record:")
indent = '    '
print(indent+"Dept:", dept)
print(indent+"Name:", name)
print(indent+"Roll No:", roll_no)
print(indent+"Library Card No:", lib_id)
```

Output

```
Student record:
    Dept: ABC
    Name: MR FNAME LNAME
    Roll No: 999
    Library Card No: AX999
```

(a)

```
ABD
XYZ
MR
FNAME LNAME
999
```

(b)

```
ABC
XYZ
MRS
FNAME LNAME
999
```

(c)

```
ABC
XYZ
MR
FNAME LNAME
999
```

(d)

```
ABC
MR
XYZ
FNAME LNAME
999
```

Answer

(c)

```
ABC
XYZ
MR
FNAME LNAME
999
```

Solution

Self explanatory.

Problem 17

```
match = False
if s.count('(') == s.count(')'):
    if s.count('[') == s.count(']'):
        if s.count('{') == s.count('}'):
            match = True
```

Question

If **s = "abcd(efgh(ijkl}{{}))"** what will be value of **match** at the end of execution and justification?

(a) False, the number of opening and closing brackets are only considered

(b) False, the position of opening and closing brackets are not considered.

(c) True, the number of opening and closing brackets are only considered

(d) True, the number and position of opening and closing brackets are considered.

Answer

(c) True, the number of opening and closing brackets are only considered

Solution

The code blocks initialize the boolean variable `match` to `False`. In outermost `if` statement `if s.count('(') == s.count(')'):`, count of opening parentheses `(` and closing parentheses `)` are compared. If this is `True`, in the intermediate `if` condition, the count of opening and closing square brackets `[` and `]` are compared. If the previous two `if` statements evaluates `True`, the innermost `if` condition executes and verifies whether the count of opening and closing braces `{` and `}` are equal. If all `if` statements evaluate `True`, the boolean variable `match` set to `True`. It only takes into consideration the number of opening and closing brackets (parentheses, square brackets and braces), not their position.