Week-1, Practice Assignment (theory)

Problem-1

Question

What will be the output of the following statement?

```python
print('15 % 3 * 2 + 2')
```

(a) `12`

(b) `2`

(c) `15 % 3 * 2 + 2`

(d) `'15 % 3 * 2 + 2'`

Answer

(c)

Solution

The `print()` function considers a mathematical expression as string when it is surrounded by quotes. The expression inside the quotes will be displayed as it is on the console. Hence, option (c) is the correct answer.

Problem-2

Question

What will be the output of the following statement?

```python
print("a + 'b' + c + '' + d")
```

(a) `a + 'b' + c + '' + d`

(b) `"a + 'b' + c + '' + d"`

(c) `a + b + c + '' + d`

(d) `a + ''b'' + c + '' + d`

Answer

(a)

Solution

The `print()` function assumes any expression inside quotes as string. Python does not allow double quotes inside of double quotes or single quotes inside of single quotes. But it does allow to print single quotes inside double quoted string and vice versa. Hence the value `a + 'b' + c + '' + d` is displayed as it is. The starting and ending quotes are not displayed. Hence, (a) is True, while (b) is False. The option (c) and (d) are incorrect as the actual input has single quotes surrounding the letter `b`.

Problem-3

Question

A snippet of code gives the following output when executed:

```
1 2 3 4 5
```

There is exactly one space between any two consecutive integers. Which of the following options correspond to the correct code snippet? It is a Multiple Select Question (MSQ).

(a)

```python
print(1 2 3 4 5)
```

(b)

```python
print('1 2 3 4 5')
```

(c)

```python
print(1)
print(2)
print(3)
print(4)
print(5)
```

(d)

```python
print(1, 2, 3, 4, 5)
```

Answer

(b), (d)

Solution

Option (b) is correct, `print()` function displays the string passed to the console as it is. When there are multiple values inside the `print()` function separated by commas, those values are concatenated together using `space` and the output is displayed. Hence, option (d) is also correct.

Problem-4

Question

What will be datatype of following expressions?

```
13 % 5 // 2 * 30 ** 5
"@Python"
20 ** 10 / 2 + 25 - 70
False
20 * 100.0 // 11 % 5
```

(a) `float`

(b) `int`

(c) `str`

(d) `bool`

Answer

1- (b); 2- (c); 3- (a); 4- (d); 5-(a)

Solution

Line-1: Option (b). The expression includes `%`, `//`, `*` and `**` operators. The output from these operators is an integer when all operands are integers.
Line-2: Option (c). It is a string. So, the type will be `str`.
Line-3: Option (a). The operator `/` is a `float` operation, this means it will give output as `float` type. Since, the expression includes the `/` operator, the output of the expression becomes `float`.
Line-4: Option (d). This is a `bool` value.
Line-5: Option (a). The expression involves a `float` value `100.0`. Hence, the entire expression is evaluated as `float`.

Problem-5

Question

How does the Python interpreter parenthesize the following expression?

```
0 ** 1 ** 2 ** 3 ** 2
```

(a) `(((0 ** 1) ** 2) ** 3) ** 2`

(b) `(0 ** (((1 ** 2) ** 3) ** 2))`

(c) `0 ** (1 ** (2 ** (3 ** 2)))`

(d) `(0 ** ((1 ** 2) ** (3 ** 2)))`

Answer

(c)

Solution

The power operator `**` has right to left associativity. Hence, option (c) is the correct way of computation.

Problem-6

Question

How does the Python interpreter parenthesize the following expression?

```python
8.2 * 10 ** 4 + 19
```

(a) `((8.2 * 10) ** 4) + 19`

(b) `8.2 * (10 ** (4 + 19))`

(c) `(8.2 * (10 ** 4)) + 19`

(d) `(8.2 * 10) ** (4 + 19)`

Answer

(c)

Solution

The order of precedence is `**` > `*` > `+`. Hence, option (c) is the correct way of parsing for computation.

Problem-7

Question

How does the Python interpreter parenthesize the following expression?

```
not False or True and False
```

(a) `not (False or (True and False))`
(b) `(not False) or (True and False)`
(c) `not ((False or True) and False)`
(d) `((not False) or True) and False`

Answer

(b)

Solution

Logical operators have precedence `not` > `and` > `or`. Therefore, option (b) is correct.

Problem-8

Question

What will be the output of the following statement?

```
not 0 and 10 // 5 == 2
```

(a) `True`

(b) `False`

Answer

(a)

Solution

Option (a) is right. The precedence of operators are `//` > `==` > `not` > `and`. Therefore, the computation follows construct: `(not 0) and ((10 // 5) == 2)`.

Problem-9

Question

Given a string variable `word` that stores some word in the English language, we wish to create a new string with just two characters:

- The first character in the new string is the first letter in `word`.
- The second character in the new string the last letter in `word`.

Assume that `word` has at least three characters. Which of the following lines can be used to create the new string? It is a Multiple Select Question (MSQ).

(a) `word[0] + word[1]`

(b) `word[0] + word[-1]`

(c) `word[0] + word[len(word) - 1]`

(d) `word[-1] + word[len(word)]`

Answer

(b), (c)

Solution

Option (b) and (c) are correct. The indexing of characters from left to right in the variable `word` starts at index `0` and ends at `len(word) - 1`. The first and last letter can be accessed using the `word[0]` and `string[len(word) - 1]` respectively. Alternatively, the indexing starts at `-1` and ends at `-len(word)` when we move from right to left in the string. Using negative indexing, the first and last letter can be retrieved by `word[-len(word)]` and `word[-1]` respectively.