1. In Python 3, what is the output of `type(range(5))`. (What data type it will return).

Hint: range() in Python

- int
- list
- **range**
- None

Explanation:

in Python 3, the `range()` function returns range object, not list.

2. What is the data type of `print(type(10))`

- float
- integer
- **int**

3. What is the data type of the following

```
aTuple = (1, 'Jhon', 1+3j)

print(type(aTuple[2:3]))
```

Refer:

- Python Data types
- tuples in Python

- list
- complex
- **tuple**

Explanation:

When we access a tuple using the subscript `atuple[start : end]` operator, it will always return a tuple. We also call it tuple slicing. (taking a subset of a tuple using the range of indexes).

4. What is the data type of `print(type(0xFF))`

- number
- hexint
- **hex**
- int

Explanation:

We can represent integers in binary, octal and hexadecimal formats.

- `0b` or `0B` for Binary and base is 2
- `0o` or `0O` for Octal and base is 8
- `0x` or `0X` for Hexadecimal and base is 16

5. What is the result of `print(type([]) is list)`

- False
- **True**

6. What is the output of `print(type({}) is set)`

- **True**
- False

Explanation:

When the object is created without any items inside the curly brackets (`{}`) then it will be created as a **dictionary** which is another built-in data structure in Python.

So whenever you wanted to create an empty set always use the `set()` constructor

7. What is the output of the following code?

```
x = 50

def fun1():

    x = 25

    print(x)



fun1()

print(x)
```

- NameError
- 25
  25
- **25**
  50

Explanation:

A variable declared outside of all functions has a GLOBAL SCOPE. Thus, it is accessible throughout the file. And variable declared inside a function is a local variable whose scope is limited to its function.

8. What is the output of the following variable assignment?

```
x = 75

def myfunc():

    x = x + 1

    print(x)



myfunc()

print(x)
```

Refer Variables in Python.

- **Error**
- 76
- 1
- None

Explanation:

Here we have not used a `global` keyword to reassign a new value to global variable `x` into `myfunc()` so Python assumes that `x` is a local variable.

It means you are accessing a local variable before defining it. that is why you received a `UnboundLocalError: local variable 'x' referenced before assignment`

The correct way to modify the global variable inside a function:

```
x = 75



def myfunc():

    global x

    x = x + 1

    print(x)



myfunc()

print(x)
```

9. Select all the right ways to create a string literal `Ault'Kelly`

- `str1 = 'Ault\\'Kelly'`
- **`str1 = 'Ault\'Kelly'`**
- **`str1 = """Ault'Kelly"""`**

10. What is the output of the following code

```
print(bool(0), bool(3.14159), bool(-3), bool(1.0+1j))
```

- False True False True
- True True False True
- True True False True
- **False True True True**

Explanation:

- If we pass A zero value to `bool()` construtor, it will treat as a boolean `False`
- Any non-zero value is boolean `True`

11. Please select the correct expression to reassign a global variable "**x**" to 20 inside a function `fun1()`

```
x = 50

def fun1():

    # your code to assign global x = 20

fun1()

print(x) # it should print 20
```

- `global x =20`

```
global var x

x = 20
```

- `global.x = 20`

```
global x

x = 20
```

12. Select all the valid String creation in Python

-
- `str1 = 'str1'`
- `str1 = "str1"`
- `str1 = '''str'''`
-
- `str1 = 'str1'`
- `str1 = "str1""`
- `str1 = '''str1''`
- `str1 = str(Jessa)`

Explanation:

Strings in Python are surrounded by either single quotation marks or double quotation marks. Also, You can create a multiline string using three quotation marks.

13. What is the output of the following code

```
def func1():

    x = 50

    return x

func1()

print(x)
```

- 50
- **NameError**
- None
- 0

Explanation:

You will get a `NameError: name 'x' is not defined`. To access the function's return value we must accept it using an assignment operator like this

```
def myfunc():

    x = 50

    return x

x = myfunc()

print(x)
```

1. What is the output of the following Python code

```
x = 10

y = 50

if x ** 2 > 100 and y < 100:

    print(x, y)
```

- 100 500
- 10 50
- **None**

2. Which of the following operators has the highest precedence?

Hint: Python operators precedence

- `not`
- `&`
- `*`
- `+`

3. What is the output of the following addition (`+`) operator

```
a = [10, 20]
```

```
b = a

b += [30, 40]

print(a)

print(b)
```

- **[10, 20, 30, 40]**
  [10, 20, 30, 40]
- [10, 20]
  [10, 20, 30, 40]

Explanation:

Because both `b` and `a` refer to the same object, when we use addition assignment `+=` on `b`, it changes both `a` and `b`

4. What is the output of the expression `print(-18 // 4)`

- -4
- 4
- **-5**
- 5

Explanation:

In the case of **floor division** operator (`//`), when the result is negative, the result is rounded down to the next smallest (big negative) integer.

5. What is the output of `print(10 - 4 * 2)`

- **2**
- 12

Explanation:

The multiplication(*) operator has higher precedence than minus(-) operator

6. What is the output of the following code

```
x = 6

y = 2

print(x ** y)

print(x // y)
```

- 66
  0
- 36
  0
- 66
  3
- **36**
  3

Explanation:

- The Exponent `(**)` operator performs exponential (power) calculation. so here `6 ** 2` means `6*6 = 36`
- The `//` is the Floor Division operator

7. **4** is `100` in binary and **11** is `1011`. What is the output of the following bitwise operators?

```
a = 4

b = 11

print(a | b)

print(a >> 2)
```

- **15**
  1

- 14
  1

Explanation:

Bitwise right shift operator(`>>`): The a's value is moved right by the 2 bits.

8. What is the output of the following assignment operator

```
y = 10

x = y += 2

print(x)
```

- 12
- 10
- **SynatxError**

Explanation: `x = y += 2` expression is Invalid

9. What is the output of `print(2 ** 3 ** 2)`

- **64**
- 512

Explanation:

Remember, we have not used brackets here. And Exponent operator `**` has right-to-left associativity in Python.

10. What is the value of the following Python Expression

```
print(36 / 4)
```

- **9.0**
- 9

Explanation:

Remember the result of a **division operator(`/`)**, is always float value.

11. Bitwise shift operators (`<<`, `>>`) has higher precedence than Bitwise And(`&`) operator

- False
- **True**

12. What is the output of the following code

```
x = 100

y = 50

print(x and y)
```

- True
- 100
- False
- **50**

Explanation:

In Python, When we join two non-Boolean values using a `and` operator, the value of the expression is the second operands, not `True` or `False`.

13. What is the output of `print(2 * 3 ** 3 * 4)`

- **216**
- 864

Explanation:

The exponent (`**`) operator has higher precedence than multiplication (`*`). Therefore the statement `print(2 * 3 ** 3 * 4)` evaluates to `print(2 * 27 * 4)`

14. What is the output of the following code

```
print(bool(0), bool(3.14159), bool(-3), bool(1.0+1j))
```

- True True False True
- **False True True True**
- True True False True
- False True False True

Explanation:

- If we pass A zero value to `bool()` constructor, it will treat it as a boolean `False`.
- Any non-zero value will be treated as a boolean `True`.

15. What is the output of `print(2%6)`

- ValueError
- 0.33
- **2**

Explanation:

The first number is the numerator, and the second is the denominator. here, 2 is divided by 6. So the remainder is 2. Therefore the result is 2