1. The given code snippet is a recursive function that implements the Euclidean algorithm to find the greatest common divisor (GCD) of two numbers. Let's walk through the function step by step to understand how it works and what the output will be.

   Here's the code snippet again:

```
 def func(a, b):
return b if a == 0 else func(b % a, a)

print(func(30, 75))
```

First Call: func(30, 75)`

Since `a` is not 0 (`30 != 0`), it goes to the `else` part and calls `func(b % a, a)`.

b % a = 75 % 30 = 15`, so it becomes `func(15, 30)`.

Second Call: `func(15, 30)`

Again, `a` is not 0 (`15 != 0`), it goes to the `else` part and calls `func(b % a, a)`.
b % a = 30 % 15 = 0`, so it becomes `func(0, 15)`.

Third Call: `func(0, 15)`

This time, `a` is 0 (`0 == 0`), so it returns `b`, which is `15`.

Now, the recursion unwinds and returns the result from the innermost call to the outermost call:

func(0, 15)` returns `15`.
func(15, 30)` returns `15` (since `func(0, 15)` returned `15`).
func(30, 75)` returns `15` (since `func(15, 30)` returned `15`).

Therefore, the output of the code snippet is: (c) 15

1. def func(a, b):

```
return b if a == 0 else func(b % a, a)
print(func(30, 75))
```

This function computes the GCD of 30 and 75 using the Euclidean algorithm. As derived the output is:

(c). 15

2. Type of `even_numbers`

```
numbers = (4, 7, 19, 2, 89, 45, 72, 22)

sorted_numbers = sorted(numbers)
```

```
even = lambda a: a % 2 == 0
even_numbers = filter(even, sorted_numbers)
print(type(even_numbers))
```

The `filter` function returns a filter object:

(b). Filter

3. Data Type of `*args`

When passed into a function, `*args` is stored as a:

(a). Tuple

4. Length of Combined Sets

```
set1 = {14, 3, 55}
set2 = {82, 49, 62}
set3 = {99, 22, 17}
print(len(set1 + set2 + set3))
```

You cannot use the `+` operator on sets. This results in an error:

(d). Error

5. Keyword to Raise Exceptions

In Python, the keyword used to raise exceptions is:

(a). raise

6. Module for Date-Time Computations

The module needed for date-time computations is:

(c). datetime

7. Output of the Code Snippet

print(4**3 + (7 + 5)**(1 + 1))

This evaluates to \(4^3 + (7+5)^{(1+1)} = 64 + 144 = 208\):

(c). 208

8. Function to Convert Date to Corresponding Time

Both `strptime` and `strftime` handle date and time conversions:

(c). both a) and b)**

9. Nature of Python Tuple

A Python tuple is:

(b). immutable

10. Built-in Function for Range of Integers

The built-in function that returns a range object is:

(a). range()

11. Function Without a Name

A function without a name is a:

(c). Lambda function

12. Purpose of Pickle Module

The module Pickle is used for:

(c). Both A and B (Serializing and de-serializing Python object structure)

13. Method to Convert Python Objects for Binary Files

The method to convert Python objects for writing data in a binary file is:

(b). dump() method

14. Method to Unpickle Data from a Binary File

The method used to unpickle data from a binary file is:

(a). load()

15. Content of a Text File

A text file contains:

(d). All of the mentioned above (Alphabets, Numbers, Special symbols)

16. Code to Get Specified Output

To get the specified output from the dictionary `captains`, both options (a) and (b) are correct:

(d) both a and b

17. Create an Empty Dictionary

The line of code that creates an empty dictionary is:

(d) captains = {}

18. Adding Data to Dictionary

To add key-value pairs to the dictionary, the correct code snippet is:

(b) captains["Enterprise"] = "Picard"

captains["Voyager"] = "Janeway"

captains["Defiant"] = "Sisko"

19. Display Ship and Captain Names with Context

The code that provides additional context correctly is:

(b) for ship, captain in captains.items():**

print(f"The {ship} is captained by {captain}.")

20. Remove an Entry from the Dictionary

To remove the entry for the key "Discovery":

(c) del captains["Discovery"]