1 What will be the output of the following code snippet?

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

a) 10

b) 20

c) 15

d) 0

ans:
The output of the code snippet is 15.

The function `fun` takes two arguments, `a` and `b`. If `a` is equal to 0, the function returns `b`. Otherwise, the function calls itself recursively with the arguments `b % a` and `a`. In this case, the function will be called recursively with the arguments `15` and `30`. The first time the function is called recursively, b % a is equal to 15, which is not equal to 0. So the function will call itself again with the arguments `15` and `30`. The second time the function is called recursively , b % a is equal to 0, so the function will return `15`.

The following is the execution of the code snippet:

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

print(fun(30, 75))
```

- The first time the function is called, the arguments are `30` and `75`.
- The function checks if `a` is equal to 0. Since `a` is not equal to 0, the function calls itself recursively with the arguments `b % a` and `a`.
- The second time the function is called, the arguments are `15` and `30`.
- The function checks if `a` is equal to 0. Since `a` is not equal to 0, the function calls itself recursively with the arguments `b % a` and `a`.
- The third time the function is called, the arguments are `15` and `30`.
- The function checks if `a` is equal to 0. Since `a` is equal to 0, the function returns `b`. The value of `b` is 15.
- The function returns the value of `15` to the second call of the function.

- The second call of the function returns the value of `15` to the first call of the function.
- The first call of the function returns the value of `15` to the main program.
- The main program prints the value of `15`.

Therefore, the output of the code snippet is 15

2 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))

a) Int

b) Filter

c) List

d) Tuple

ans:

The `filter()` function in Python returns an iterator, which is an object that can be used to iterate over a sequence of elements. The `list()` function can be used to convert an iterator to a list.

In the code snippet, the `filter()` function returns an iterator that contains all the even numbers from the sorted list `sorted`. The `list()` function then converts the iterator to a list and assigns it to the variable `even_numbers`.

The type of the variable `even_numbers` is a list. So the answer is c) List.

Here is a breakdown of the code snippet:

Python
```
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 first line defines a list called `numbers` that contains 8 integers.
- The second line sorts the list `numbers` and assigns the result to the variable `sorted_numbers`.

- The third line defines a lambda function called `even` that takes an integer as input and returns True if the integer is even, and False otherwise.
- The fourth line uses the `filter()` function to create an iterator that contains all the even numbers from the sorted list `sorted_numbers`.
- The fifth line uses the `list()` function to convert the iterator to a list and assigns it to the variable `even_numbers`.
- The sixth line prints the type of the variable `even numbers`.

The output of the code snippet is:

```
<class 'list'>
```

Therefore, the type of the variable `even numbers` is a list.

3) As what datatype are the *args stored, when passed into

a) Tuple

b) List

c) Dictionary

d) none

ans:

The asterisk (*) in front of the parameter name `args` in the function definition tells Python that the function can accept a variable number of arguments. These arguments are stored in a tuple, which is a sequence of immutable objects.

For example, the following code snippet defines a function called `my_function` that takes a variable number of arguments:

```
def my_function(*args):
  print(args)

my_function(1, 2, 3, 4, 5)
```

When the function `my_function` is called with the arguments `1`, `2`, `3`, `4`, and `5`, the arguments are stored in the tuple `args`. The `print()` statement in the function then prints the contents of the tuple `args`, which is `[1, 2, 3, 4, 5]`.

Therefore, the datatype of the arguments passed into `*args` is a tuple.

4) set1 = {14, 3, 55}

set2 = {82, 49, 62}

set3={99,22,17}

print(len(set1 + set2 + set3))

a) 105

b) 270

c) 0

d) Error

ans: The code snippet tries to add three sets together using the `+` operator. However, the `+` operator can only be used to add two sequences together. Sets are not sequences, so the `+` operator cannot be used to add them together.

The correct way to add two sets together is to use the `+` operator. The `+` operator returns a new set that contains all the elements of the two sets.

The following code snippet shows how to add two sets together using the `|` operator:

Python
```
set1 = {14, 3, 55}
set2 = {82, 49, 62}
set3={99,22,17}

new_set = set1 | set2 | set3

print(len(new_set))
```

The output of the code snippet is:

```
105
```

Therefore, the answer to the first question is d) Error.

The answer to the second question is `in`.

The keyword `in` is used to check if an element is present in a sequence. For example, the following code snippet checks if the number 14 is present in the set `set1`:

```
set1 = {14, 3, 55}

if 14 in set1:
  print("14 is present in the set")
else:
  print("14 is not present in the set")
```

The output of the code snippet is:

```
14 is present in the set
```

Therefore, the keyword `in` is used to check if an element is present in a sequence.

5) What keyword is used in Python to raise exceptions?

a) raise

b) try

c) goto

d) except

ans:
The `raise` keyword is used to raise an exception in Python. An exception is an event that occurs during the execution of a program that disrupts the normal flow of the program. When an exception is raised, the program stops executing the current line of code and jumps to the exception handler.

The following code snippet shows how to raise an exception in Python:

Python
```
def divide_numbers(x, y):
  if y == 0:
    raise ValueError("Division by zero")
  else:
    return x / y

try:
  result = divide_numbers(10, 0)
except ValueError as e:
  print(e)

print(result)
```

The `divide_numbers()` function raises a `ValueError` exception if the second argument is equal to 0. The `try` block tries to execute the function. If the exception is raised, the control flow jumps to the `except` block. The `except` block prints the error message from the exception. The `print()` statement at the end of the code snippet prints the value of the variable `result`, which is `None` because the `divide_numbers()` function raised an exception.

Therefore, the keyword `raise` is used to raise exceptions in Python.

6) Which of the following modules need to be imported to handle date time computations in Python?

a) timedate

b) date

c) datetime

d) time

ans: To handle date and time computations in Python, you need to import the **`datetime`** module.

So, the correct answer is: c) datetime

7) What will be the output of the following code snippet?

print(4**3 + (7 + 5)**(1 + 1)) a) 248 b) 169 c) 208 d) 233

ans:

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

- The first line evaluates the expression `4**3` and assigns the result to the variable `a`.
- The second line evaluates the expression `(7 + 5)**(1 + 1)` and assigns the result to the variable `b`.
- The third line prints the sum of the variables `a` and `b`.

The output of the code snippet is: 208

8) Which of the following functions converts date to corresponding time in Python?

a) strptime

b) strftime

c) both a) and b)

d) None

ans:
The answer is c) both a) and b).

The `strptime()` function in Python converts a string to a datetime object. The `strftime()` function in Python converts a datetime object to a string. Both functions can be used to convert a date to a corresponding time.

The `strptime()` function takes a string as input and a format string as a second argument. The format string specifies how the string should be parsed. The `strftime()` function takes a datetime object as input and a format string as a second argument. The format string specifies how the datetime object should be formatted.

9) The python tuple is _____ in nature.
a) mutable
b)immutable
c)unchangeable
d) none

ans: The correct answer is b) immutable.

Tuples in Python are immutable, which means that their elements cannot be changed once they are created. This is in contrast to lists, which are mutable and can be changed after they are created.

10) The ___ is a built-in function that returns a range object that consists series of integer numbers, which we can iterate using a for loop.

A. range()
B. set()
C. dictionary{}
D. None of the mentioned above

Ans: The answer is A. range().

The `range()` function in Python is a built-in function that returns a range object that consists of a series of integer numbers.

The range object can be iterated using a for loop.

11 Amongst which of the following is a function which does not have any name?

A. Del function

B. Show function

C. Lambda function

D. None of the mentioned above

Ans:
The correct answer is C. Lambda function.

A lambda function is a small anonymous function that can be created without a name. Lambda functions are often used for short, one-off tasks, such as filtering or mapping a sequence of elements.

12 The module Pickle is used to ___.

A. Serializing Python object structure

B. De-serializing Python object structure

C. Both A and B

D. None of the mentioned above

Ans:
The correct answer is C. Both A and B.

The pickle module in Python is used to serialize and deserialize Python object structures. Serialization is the process of converting a Python object structure into a byte stream. Deserialization is the process of converting a byte stream back into a Python object structure.

The pickle module can be used to save Python objects to a file or to a database. It can also be used to send Python objects over a network.

13 Amongst which of the following is / are the method of convert Python objects for writing data in a binary file?

A. set() method

B. dump() method

C. load() method

D. None of the mentioned above

Ans:
The answer is B. dump() method.

The `dump()` method in the pickle module is used to serialize Python objects and write them to a binary file. The `load()` method in the pickle module is used to deserialize Python objects from a binary file.

14 Amongst which of the following is / are the method used to unpickling data from a binary file?

A. load()

B. set() method

C. dump() method

D. None of the mentioned above

Ans:
The answer is A. load().

The `load()` method in the pickle module is used to deserialize Python objects from a binary file. The `dump()` method in the pickle module is used to serialize Python objects and write them to a binary file. The `set()` method in the Python standard library is used to create a set, which is a collection of unique elements. The `set()` method is not used to unpickle data from a binary file.

Therefore, the correct answer is A. load().

15. A text file contains only textual information consisting of ___.

A. Alphabets

B. Numbers

C. Special symbols

D. All of the mentioned above

Ans: The answer is D. All of the mentioned above.

A text file is a file that contains only textual information. This textual information can consist of alphabets, numbers, and special symbols.

The following are some examples of text files:

- A file that contains a poem
- A file that contains a story
- A file that contains a code
- A file that contains a list of names

Text files are often used to store data that is human-readable. This data can be read and understood by humans without the need for special software.

16 Which Python code could replace the ellipsis (...) below to get the following output? (Select all that apply.)

```
captains = {
    "Enterprise": "Picard",
    "Voyager": "Janeway",
    "Defiant": "Sisko",
}
```

Enterprise Picard,

Voyager Janeway

Defiant Sisko

a) for ship, captain in captains.items(): print(ship, captain)

b) for ship in captains: print(ship, captains[ship])

c) for ship in captains: print(ship, captains)

d) both a and b

ans; The code snippet `a` uses the `items()` method to iterate over the dictionary `captains`. The `items()` method returns a list of tuples, where each tuple contains the key and value of a dictionary entry. The code snippet then prints the first element of each tuple, which is the key, and the second element of each tuple, which is the value.

The code snippet `b` uses the `keys()` method to iterate over the dictionary `captains`. The `keys()` method returns a list of the keys of a dictionary. The code snippet then prints each key and the value associated with that key.

Therefore, both code snippets `a` and `b` are correct.

17) Which of the following lines of code will create an empty dictionary named captains?

a) captains = {dict}

b) type(captains)

c) captains.dict()

d) captains = {}

ans:
The answer is d) captains = {}.

The `{}` syntax is used to create an empty dictionary in Python. The other options are not valid ways to create an empty dictionary.

Here is a breakdown of the different options:

- Option `a`: The `{dict}` syntax is not valid. The `dict` keyword is used to create a dictionary object, but it cannot be used to create an empty dictionary.
- Option `b`: The `type(captains)` function returns the type of the variable `captains`. The variable `captains` does not exist yet, so the `type(captains)` function will return the type `NoneType`.
- Option `c`: The `captains.dict()` method is not a valid method. The `dict()` method is a method of the `collections` module, but it cannot be used to create an empty dictionary.
- Option `d`: The `{}` syntax is used to create an empty dictionary. This is the correct option.

18) Now you have your empty dictionary named captains. It’s time to add some data!

Specifically, you want to add the key-value pairs "Enterprise": "Picard", "Voyager": "Janeway", and "Defiant": "Sisko".

Which of the following code snippets will successfully add these key-value pairs to the existing captains dictionary?

a) captains{"Enterprise" = "Picard"} captains{"Voyager" = "Janeway"} captains{"Defiant" = "Sisko"}

b) captains["Enterprise"] = "Picard" captains["Voyager"] = "Janeway" captains["Defiant"] = "Sisko"

c) captains = {

"Enterprise": "Picard", "Voyager": "Janeway", "Defiant": "Sisko",

}

d) None of the above

ans: The answer is b) captains["Enterprise"] = "Picard" captains["Voyager"] = "Janeway" captains["Defiant"] = "Sisko".

The other options are not valid ways to add key-value pairs to a dictionary.

Here is a breakdown of the different options:

- Option `a`: The syntax `captains{"Enterprise" = "Picard"} captains{"Voyager" = "Janeway"} captains{"Defiant" = "Sisko"}` is not valid. The `captains` variable is not a dictionary yet, so it cannot be assigned to a dictionary.
- Option `c`: The syntax `captains = { "Enterprise": "Picard", "Voyager": "Janeway", "Defiant": "Sisko", }` is not valid. The `captains` variable is already defined as an empty dictionary. The code snippet tries to redefine the `captains` variable as a dictionary with three key-value pairs. This is not allowed.
- Option `b`: The syntax `captains["Enterprise"] = "Picard" captains["Voyager"] = "Janeway" captains["Defiant"] = "Sisko"` is valid. This code snippet successfully adds the three key-value pairs to the existing `captains` dictionary.

19) You're really building out the Federation Starfleet now! Here's what you have:

captains = {

"Enterprise": "Picard",

"Voyager": "Janeway",

"Defiant": "Sisko", "Discovery": "unknown",

}Now, say you want to display the ship and captain names contained in the dictionary, but you also want to provide some additional context. How could you do it?

a) for item in captains.items(): print(f"The [ship] is captained by [captain].")

b) for ship, captain in captains.items(): print(f"The {ship} is captained by {captain}.")

c) for captain, ship in captains.items(): print(f"The {ship} is captained by {captain}.")

d) All are correct

ans: The answer is d) All are correct.

The code snippets `a`, `b`, and `c` will all display the ship and captain names contained in the dictionary `captains`, but with some additional context.

The code snippet `a` uses the `items()` method to iterate over the dictionary `captains`. The `items()` method returns a list of tuples, where each tuple contains the key and value of a dictionary entry. The code snippet then uses the `format()` method to format the output string. The `format()` method takes a string and a sequence of values as input, and it returns a new string where the values are substituted into the string.

The code snippet `b` uses the `items()` method to iterate over the dictionary `captains`. The `items()` method returns a list of tuples, where each tuple contains the key and value of a dictionary entry. The code snippet then uses the `f-string` syntax to format the output string. The `f-string` syntax is a new feature in Python 3.6 that allows you to embed expressions inside strings.

The code snippet `c` uses the `items()` method to iterate over the dictionary `captains`. The `items()` method returns a list of tuples, where each tuple contains the key and value of a dictionary entry. The code snippet then uses the `format()` method to format the output string. However, the code snippet reverses the order of the key and value in the tuple.

Therefore, all three code snippets are correct and will produce the same output.

20 You've created a dictionary, added data, checked for the existence of keys, and iterated over it with a for loop. Now you're ready to delete a key from this dictionary:

```
captains = {

"Enterprise": "Picard",

"Voyager": "Janeway",

 "Defiant": "Sisko", "Discovery": "unknown",

}
```

What statement will remove the entry for the key "Discovery"?

a) del captains

b) captains.remove()

c) del captains["Discovery"]

d) captains["Discovery"].pop()

ans: `The` answer is c) del captains["Discovery"].

The other options are not valid ways to delete a key from a dictionary.

Here is a breakdown of the different options:

- Option `a`: The `del captains` statement will delete the entire dictionary `captains`. This is not what we want to do, because we only want to delete the entry for the key "Discovery".
- Option `b`: The `captains.remove()` method does not exist. There is no method called `remove()` in the `dict` class.
- Option `c`: The `del captains["Discovery"]` statement will delete the entry for the key "Discovery" from the dictionary `captains`. This is the correct option.
- Option `d`: The `captains["Discovery"].pop()` method will also delete the entry for the key "Discovery" from the dictionary `captains`. However, the `pop()` method returns the value of the deleted entry, while the `del` statement does not return anything.