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 code snippet provided is an implementation of the **Euclidean algorithm** for finding the greatest common divisor (GCD) of two numbers. Here's a step-by-step breakdown of how the function works:

Code Analysis

python
Copy code

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

This function takes two arguments, `a` and `b`, and uses recursion to compute the GCD:

1. **Base Case:** If `a` is 0, the function returns `b`. This is because the GCD of 0 and any number `b` is `b`.
2. **Recursive Case:** If `a` is not 0, the function recursively calls itself with the arguments `b % a` and `a`. This is based on the principle that GCD(a, b) is equivalent to GCD(b % a, a).

Execution

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

1. **First Call:** `func(30, 75)`
    - `a` is not 0, so it proceeds to `func(75 % 30, 30)`
    - Calculate `75 % 30` which is `15`.
    - Now, `func(15, 30)`.
2. **Second Call:** `func(15, 30)`
    - `a` is not 0, so it proceeds to `func(30 % 15, 15)`
    - Calculate `30 % 15` which is `0`.
    - Now, `func(0, 15)`.
3. **Third Call:** `func(0, 15)`
    - `a` is 0, so it returns `15`.

The final result of the recursive calls is `15`.

Conclusion

The output of the code snippet `print(func(30, 75))` is `15`.

Correct Answer

c) 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-

To determine the output of the code snippet, we need to understand the behavior of the `filter` function in Python.

Code Breakdown

python
Copy code

```
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))
```

Steps

Define the Tuple:
python
Copy code

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

1.

Sort the Tuple:
python

Copy code
```
sorted_numbers = sorted(numbers)
```

2. This results in: `[2, 4, 7, 19, 22, 45, 72, 89]`.

Define the Lambda Function:
python
Copy code
```
even = lambda a: a % 2 == 0
```

3. This lambda function checks if a number is even.

Filter the Sorted List:
python
Copy code
```
even_numbers = filter(even, sorted_numbers)
```

4. `filter` returns an iterator (specifically, a filter object) that yields items from `sorted_numbers` for which the `even`function returns `True`.

Check the Type of the Filter Object:
python
Copy code
```
print(type(even_numbers))
```

5.

Conclusion

The `filter` function in Python returns a `filter` object, which is an iterator, not a list, tuple, or integer. Therefore, the output of `print(type(even_numbers))` will be `<class 'filter'>`.

Correct Answer

b) Filter

3) As what datatype are the *args stored, when passed into a) Tuple
b) List
c) Dictionary

d) none

Ans-

When using `*args` in a Python function, it allows the function to accept an arbitrary number of positional arguments. These arguments are stored in a tuple.

Here's a brief explanation:

Example

python
Copy code

```
def example_function(*args):
    print(type(args))

example_function(1, 2, 3)
```

Explanation

- `*args` allows you to pass a variable number of arguments to the function.
- Inside the function, `args` is stored as a **tuple**.

When you call `example_function(1, 2, 3)`, `args` becomes `(1, 2, 3)`, which is a tuple.

Conclusion

Therefore, `*args` are stored as a **tuple** when passed into a function.

Correct Answer

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 given code snippet attempts to concatenate sets, which is not directly supported in Python. Here's a step-by-step analysis:

Code Analysis

python

Copy code

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

Explanation

1. **Set Initialization:**
   - `set1 = {14, 3, 55}`
   - `set2 = {82, 49, 62}`
   - `set3 = {99, 22, 17}`
2. Each of these variables is a set, containing distinct elements.
3. **Attempted Concatenation:**
   - `set1 + set2 + set3`
4. Sets in Python do not support the `+` operator for concatenation. Instead, sets use methods like `.union()` or the `|`operator to combine sets. Trying to use `+` with sets will result in a `TypeError`.
5. **Error Handling:**
   - When executing `set1 + set2 + set3`, Python will raise a `TypeError` because the `+` operator is not defined for sets.

Conclusion

Given the attempted operation is invalid, Python will raise a `TypeError`.

Correct Answer

d) Error

5) What keyword is used in Python to raise exceptions? a) raise
b) try
c) goto

d) except

Ans-

In Python, the keyword used to raise exceptions is `raise`. Here's a brief explanation of each option for clarity:

Explanation of Keywords

`raise`: This keyword is used to throw an exception manually. It allows you to specify the exception type and message.
python
Copy code
```
raise ValueError("This is an error message")
```

-

`try`: This keyword is used to start a block of code that will be monitored for exceptions. It must be followed by at least one `except` or `finally` block.
python
Copy code
```
try:
    # Code that might raise an exception
except Exception as e:
    # Handle the exception
```

-
- **`goto`**: Python does not have a `goto` keyword. This option is not applicable in Python.

`except`: This keyword is used to catch exceptions that occur within the `try` block. It specifies how to handle the exception.
python
Copy code
```
try:
    # Code that might raise an exception
except ValueError as e:
    # Handle the ValueError exception
```

-

Conclusion

To manually trigger an exception in Python, you use the `raise` keyword.

Correct Answer

a) raise

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, the correct module to import is `datetime`. Here's a quick rundown of the options provided:

Options Breakdown

a) `timedate`:

- **Incorrect**. This is not a standard or recognized module in Python for date and time computations.

b) `date`:

- **Incorrect**. While `datetime` includes a `date` class, there is no separate `date` module in the Python Standard Library.

c) `datetime`:

Correct. This module provides classes and functions for working with dates and times, including date arithmetic, formatting, and parsing.
python
Copy code
```
import datetime
```

-

d) `time`:

Partially correct. The `time` module provides functions for working with time, like measuring time intervals and sleeping, but it does not handle dates in the same way `datetime` does. It's more focused on time-related functions rather than full date and time manipulations.
python
Copy code
```
import time
```

-

Conclusion

The `datetime` module is the standard choice for comprehensive date and time computations in Python.

Correct Answer

c) `datetime`

Example Usage:

python
Copy code

```
import datetime

# Getting the current date and time
now = datetime.datetime.now()
print(now)

# Creating a specific date
date = datetime.date(2024, 6, 13)
print(date)

# Calculating a timedelta
delta = datetime.timedelta(days=5)
new_date = date + delta
print(new_date)
```

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-

Let's break down the code snippet to understand its output:

Code Snippet

python

Copy code

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

Step-by-Step Calculation

1. **Evaluate the Exponentiation:**
   - `4**3` means 4343, which is 4×4×4=644×4×4=64.
2. **Evaluate the Inner Parentheses:**
   - `(7 + 5)` means 7+5=127+5=12.
3. **Evaluate the Outer Exponentiation:**
   - `(1 + 1)` means 1+1=21+1=2.
   - So, `(7 + 5)**(1 + 1)` means 122122, which is 12×12=14412×12=144.
4. **Addition:**
   - Now add the two results: 64+144=20864+144=208.

Conclusion

The final result of the expression `4**3 + (7 + 5)**(1 + 1)` is `208`.

Correct Answer

c) 208

Calculation Breakdown:

1. 43=6443=64
2. 7+5=127+5=12
3. 122=144122=144
4. 64+144=20864+144=208

Here's how the code works in Python:

python

Copy code

```
# Calculation

result = 4**3 + (7 + 5)**(1 + 1)

print(result)  # Output: 208
```

Therefore, the output of the code snippet is `208`.

Question 8

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

Options:

- **a) `strptime`**
- **b) `strftime`**
- **c) both a) and b)**
- **d) None**

Answer:

- **b) `strftime`**

Explanation:

- **`strftime` (String Format Time)**:
    - Converts a `datetime` object to a string according to a specified format.
    - Used for formatting dates and times as strings.

Example:
python
Copy code

```
import datetime

now = datetime.datetime.now()

formatted_time = now.strftime("%Y-%m-%d %H:%M:%S")

print(formatted_time)  # Outputs the current date and time as a formatted string
```

    -

- **`strptime` (String Parse Time)**:
    - Converts a string representation of a date/time to a `datetime` object according to a specified format.
    - Used for parsing strings to create `datetime` objects.

Example:
python
Copy code

```
import datetime

date_str = "2024-06-13 14:55:30"
```

```
dt = datetime.datetime.strptime(date_str, "%Y-%m-%d %H:%M:%S")

print(dt)  # Outputs a datetime object
```

-

Summary:

- `strftime` converts a date or time to a string format.
- `strptime` converts a string format to a date or time.

Correct Answer: b) `strftime`

Question 9

The Python tuple is _____ in nature.

Options:

- **a) mutable**
- **b) immutable**
- **c) unchangeable**
- **d) none**

Answer:

- **b) immutable**

Explanation:

- **Mutable** means that the object can be changed after its creation.
- **Immutable** means that the object cannot be changed after its creation.
- **Unchangeable** is synonymous with immutable but not typically used in Python terminology.

Python Tuples:

- Tuples are immutable, meaning that once a tuple is created, its elements cannot be changed, added, or removed.
- You can create a new tuple by combining existing tuples, but you cannot modify the original one.

Example:

python

Copy code

```
my_tuple = (1, 2, 3)

# This will raise an error: TypeError: 'tuple' object does not support item assignment

my_tuple[0] = 10
```

Correct Answer: b) immutable

Which module to import for date and time computations?

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

Options:

- **a) timedate**
- **b) date**
- **c) datetime**
- **d) time**

Answer:

- **c) datetime**

Explanation:

- **`datetime`**: This module is the primary module for handling date and time computations. It provides classes for working with both dates and times and includes features for arithmetic operations, parsing, and formatting.
- **`time`**: Although the `time` module provides time-related functions, it is not used for date computations.

Example:

python

Copy code

```
import datetime

now = datetime.datetime.now()
```

```
print(now)  # Outputs the current date and time
```

Correct Answer: c) datetime

Question 10

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

Options:

- **A. `range()`**
- **B. `set()`**
- **C. `dictionary{}`**
- **D. Noneofthementionedabove**

Answer:

A. `range()`

Explanation: The `range()` function returns a sequence of numbers, which can be iterated over in a `for` loop. It's commonly used to generate a series of numbers for iteration.

Example:

python

Copy code

```
for i in range(5):
    print(i)
```

Question 11

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

Options:

- **A. Delfunction**
- **B. Show function**

- **C. Lambda function**
- **D. Noneofthementionedabove**

Answer:

C. Lambda function

Explanation: A lambda function is an anonymous function in Python, defined using the `lambda` keyword. It does not have a name and is typically used for short, throwaway functions.

Example:

python

Copy code

```
square = lambda x: x**2

print(square(4))  # Output: 16
```

Question 12

The module Pickle is used to ___.

Options:

- **A. SerializingPythonobjectstructure**
- **B. De-serializing Python object structure**
- **C. BothAandB**
- **D. Noneofthementionedabove**

Answer:

C. Both A and B

Explanation: The `pickle` module is used for serializing (converting a Python object into a byte stream) and deserializing (converting a byte stream back into a Python object) Python object structures.

Example:

python

Copy code

```
import pickle


# Serialization

data = {'key': 'value'}

serialized_data = pickle.dumps(data)


# Deserialization

deserialized_data = pickle.loads(serialized_data)

print(deserialized_data)  # Output: {'key': 'value'}
```

Question 13

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

Options:

- **A. `set()` method**
- **B. `dump()` method**
- **C. `load()` method**
- **D. Noneofthementionedabove**

Answer:

B. `dump()` method

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

Example:

python

Copy code

```
import pickle


data = {'key': 'value'}

with open('data.pkl', 'wb') as file:

    pickle.dump(data, file)
```

Question 14

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

Options:

- **A. `load()`**
- **B. `set()` method**
- **C. `dump()` method**
- **D. Noneofthementionedabove**

Answer:

A. `load()`

Explanation: The `load()` method from the `pickle` module is used to read serialized Python objects from a binary file and deserialize them.

Example:

python

Copy code

```
import pickle


with open('data.pkl', 'rb') as file:

    data = pickle.load(file)

print(data)  # Output: {'key': 'value'}
```

Question 15

A text file contains only textual information consisting of ___.

Options:

- **A. Alphabets**
- **B. Numbers**
- **C. Special symbols**
- **D. Allofthementionedabove**

Answer:

D. Allofthementionedabove

Explanation: A text file can contain alphabets, numbers, and special symbols, as it is composed of readable characters.

Example:

plaintext

Copy code

```
Hello, World! 123
```

Question 16

Which Python code could replace the ellipsis to get the following output?

Desired Output:

Copy code

```
Enterprise Picard, Voyager Janeway, Defiant Sisko
```

Options:

a)
python

Copy code

```
for ship, captain in captains.items():

    print(ship, captain)
```

-

b)
python
Copy code

```
for ship in captains:

    print(ship, captains[ship])
```

-

c)
python
Copy code

```
for ship in captains:

    print(ship, captains)
```

-
- **d) Both a and b**

Answer:

d) Both a and b

Explanation:

- Option **a)** and **b)** correctly iterate over the dictionary and print each ship and its captain.
- Option **c)** will not provide the desired output because it prints the dictionary `captains` instead of the captain's name.

Question 17

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

Options:

- **a)** `captains = {dict}`
- **b)** `type(captains)`

- **c)** `captains.dict()`
- **d)** `captains = {}`

Answer:

d) `captains = {}`

Explanation: Creating an empty dictionary is done using curly braces `{}`. Other options do not correctly initialize an empty dictionary.

Example:

python

Copy code

```
captains = {}
```

Question 18

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

Options:

a)
python
Copy code
```
captains{"Enterprise" = "Picard"}

captains{"Voyager" = "Janeway"}

captains{"Defiant" = "Sisko"}
```

-

b)
python
Copy code
```
captains["Enterprise"] = "Picard"

captains["Voyager"] = "Janeway"
```

```
captains["Defiant"] = "Sisko"
```

-

c)
python
Copy code

```
captains = {

    "Enterprise": "Picard",

    "Voyager": "Janeway",

    "Defiant": "Sisko",

}
```

-
- **d) None of the above**

Answer:

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

Explanation: Option **b)** correctly adds key-value pairs to an existing dictionary using bracket notation. Option **c)**creates a new dictionary rather than modifying the existing one.

Example:

python

Copy code

```
captains = {}

captains["Enterprise"] = "Picard"

captains["Voyager"] = "Janeway"

captains["Defiant"] = "Sisko"
```

Question 19

How could you display the ship and captain names with additional context?

Options:

a)
python
Copy code
```
for item in captains.items():

    print(f"The [ship] is captained by [captain].")
```

-

b)
python
Copy code
```
for ship, captain in captains.items():

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

-

c)
python
Copy code
```
for captain, ship in captains.items():

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

-
- **d) All are correct**

Answer:

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

Explanation: Option **b)** correctly uses `ship` and `captain` in the formatted string. Option **a)** and **c)** will not produce the correct context as written.

Example:

python

Copy code

```
for ship, captain in captains.items():

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

Question 20

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

Options:

- **a) `del captains`**
- **b) `captains.remove()`**
- **c) `del captains["Discovery"]`**
- **d) `captains["Discovery"].pop()`**

Answer:

c) `del captains["Discovery"]`

Explanation: To delete a specific entry from a dictionary, use the `del` keyword followed by the dictionary and key. Other options are incorrect for this purpose.

Example:

python

Copy code

```
del captains["Discovery"]
```