theory-answers

August 1, 2024

[4]:
```
#1. What is the difference between a function and a method in Python?

'''Function: A function is a block of reusable code that performs a specific␣
 ↪task.
It is defined using the def keyword and can be called independently.'''



#ANSWER
def add(a, b):
    return a + b

result = add(5, 3)  # Calling the function

'''Method: A method is a function that is associated with an object.
It is defined within a class and is called on instances of that class.'''
class Calculator:
    def add(self, a, b):
        return a + b

calc = Calculator()
result = calc.add(5, 3)  # Calling the method on an instance
```

[5]:
```
#2. Explain the concept of function arguments and parameters in Python.



#ANSWER
'''Parameters: Parameters are the variables listed in the function definition.
 ↪'''

def greet(name):
    return f"Hello, {name}!"
'''Arguments: Arguments are the actual values passed to the function when it is␣
 ↪called.'''
message = greet("DAKSH")
```

[6]:
```
# 3.What are the different ways to define and call a function in Python?


#ANSWER
#Standard Function Definition:
def say_hello():
    print("Hello!")

say_hello()  # Calling the function

# Function with Parameters:
def greet(name):
    print(f"Hello, {name}!")

greet("Bob")  # Calling the function with an argument

#Lambda Function:
add = lambda x, y: x + y
print(add(2, 3))  # Calling the lambda function
```

```
Hello!
Hello, Bob!
5
```

[7]:
```
# 4.What is the purpose of the return statement in a Python function?


#ANSWER
'''The return statement is used to exit a function and return a value to the␣
 ↪caller.
It can be used to return the result of a computation or simply to exit the␣
 ↪function early.'''
def square(x):
    return x * x

result = square(4)
```

[8]:
```
#5.What are iterators in Python and how do they differ from iterables?


#ANSWER
'''Iterable: An iterable is an object that can return an iterator, such as␣
 ↪lists, tuples, or dictionaries.
It supports the __iter__() method.'''
my_list = [1, 2, 3]
'''Iterator: An iterator is an object that represents a stream of data.
```

```
It supports the __next__() method and can be obtained from an iterable using␣
 ↪the iter() function.'''

my_iter = iter(my_list)
print(next(my_iter))
```

```
1
```

[9]:
```
#6.Explain the concept of generators in Python and how they are defined.

'''Generators: Generators are a type of iterable, like lists or tuples, but␣
 ↪they generate values on the fly and are memory efficient.
They are defined using the yield keyword instead of return.'''

#ANSWER
def count_up_to(max):
    count = 1
    while count <= max:
        yield count
        count += 1

for num in count_up_to(5):
    print(num)
```

```
1
2
3
4
5
```

[10]:
```
#7.What are the advantages of using generators over regular functions?
'''Memory Efficiency: Generators produce items one at a time and do not store␣
 ↪them in memory, making them more memory-efficient.
   Performance: They are more efficient for large datasets because they␣
 ↪generate items on demand rather than computing all items upfront.
   State Preservation: Generators maintain their state between yields, allowing␣
 ↪them to resume execution where they left off.'''
```

[10]:
```
'Memory Efficiency: Generators produce items one at a time and do not store them
in memory, making them more memory-efficient.\n   Performance: They are more
efficient for large datasets because they generate items on demand rather than
computing all items upfront.\n   State Preservation: Generators maintain their
state between yields, allowing them to resume execution where they left off.'
```

[11]:
```
#8.What is a lambda function in Python and when is it typically used?

#ANSWER
```

```
'''Lambda Function: A lambda function is an anonymous function defined with the␣
 ↪lambda keyword.
It is used for short, throwaway functions that are not needed elsewhere.'''
multiply = lambda x, y: x * y
print(multiply(4, 5))
'''Typical Use: Lambda functions are often used in combination with functions␣
 ↪like map(), filter(), and sorted().'''
```

```
20
```

[11]:
```
'Typical Use: Lambda functions are often used in combination with functions like
map(), filter(), and sorted().'
```

[13]:
```
#9.Explain the purpose and usage of the map() function in Python.


#ANSWER
'''Purpose: The map() function applies a given function to all items in an␣
 ↪iterable (e.g., list)
   and returns a map object (iterator).'''

numbers = [1, 2, 3, 4]
squares = map(lambda x: x * x, numbers)
print(list(squares))
```

```
[1, 4, 9, 16]
```

[14]:
```
#10.What is the difference between map(), reduce(), and filter() functions in␣
 ↪Python?


#ANSWER
'''map(): Applies a function to all items in an iterable and returns an␣
 ↪iterator.'''
nums = [1, 2, 3]
squared = map(lambda x: x**2, nums)
print(list(squared))

'''reduce(): Applies a function of two arguments cumulatively to the items of␣
 ↪an iterable,
   from left to right, reducing the iterable to a single value. It requires␣
 ↪importing from functools.
'''

from functools import reduce

nums = [1, 2, 3, 4]
```

```
result = reduce(lambda x, y: x + y, nums)
print(result)


# filter(): Filters items in an iterable based on a function that returns True␣
  ↪or False and returns an iterator.
nums = [1, 2, 3, 4]
even = filter(lambda x: x % 2 == 0, nums)
print(list(even))
```

```
[1, 4, 9]
10
[2, 4]
```

[ ]: