Up-to-date Questions and Answers from authentic resources to improve knowledge and pass the exam at very first attempt. ----- Guaranteed.

KILL EXAMS

PCAP-31-03 MCQs
PCAP-31-03 TestPrep
PCAP-31-03 Study Guide
PCAP-31-03 Practice Test
PCAP-31-03 Exam Questions

PASS

killexams.com

AICPA

PCAP-31-03

Certified Associate in Python Programming - 2025

Question: 298
What is the purpose of the __sub__() method in a Python class?

A. To define how the - operator can be used with the object.
B. To define the initial state of the object when it is created.
C. To define the methods that can be called on the object.
D. To define the attributes that the object will have.

Answer: A
Explanation: The __sub__() method in a Python class is used to define how the - operator can be used with the object. This can be useful for objects that represent values or collections that can be subtracted from each other, such as numbers or sequences.

Question: 299
Which of the following is not a valid method for the list data type in Python?

A. append()
B. insert()
C. remove()
D. divmod()

Answer: D
Explanation: divmod() is not a valid method for the list data type. It is a built-in function in Python that returns the quotient and remainder of a division operation.

Question: 300

What is the purpose of the __setitem__() method in a Python class?

A. To enable setting of individual elements of the object

B. To define the initial state of the object
C. To specify the default behavior when the object is printed
D. To enable the object to be used in mathematical operations

Answer: A

Explanation: The __setitem__() method in a Python class is used to enable setting of individual elements of the object. This method is called when you attempt to assign a value to an element of the object using square brackets, like obj[index] = value. By implementing this method, you can define custom behavior for how the object should respond to item assignment operations.

Question: 301
What is the output of the following code?

```
def func(x, y):
return x + y

func_var = func
print(func_var(2, 3))
```

A. 5
B. 6
C. TypeError: func_var() takes 2 positional arguments but 3 were given
D. NameError: name 'func_var' is not defined

Answer: A
Explanation: The func function is assigned to the variable func_var. When func_var(2, 3) is called, it invokes the func function with the arguments 2 and 3, which returns 5.

Question: 302
What is the output of the following code?

```
def foo(x):
try:
return 10 / x
except ZeroDivisionError:
return 'Cannot divide by zero'

print(foo(2))
print(foo(0))
```

A. 5.0, 'Cannot divide by zero'
B. 5.0, 0
C. 5, 'Cannot divide by zero'
D. 5.0, 'Cannot divide by zero'

Answer: D
Explanation: The foo function takes an argument x and attempts to divide 10 by x inside a try block. If a ZeroDivisionError occurs, the function returns the string 'Cannot divide by zero'. When foo(2) is called, the function returns 5.0, which is then printed. When foo(0) is called, the ZeroDivisionError is raised, and the function returns the string 'Cannot divide by zero', which is then printed. The output of the code is 5.0, 'Cannot divide by zero'.

Question: 303
What is the output of the following code?

```
a = [1, 2, 3, 4, 5]
b = a
a.remove(3)
print(b)
```

A. [1, 2, 4, 5]
B. [1, 2, 3, 4, 5]
C. [1, 2, 3, 4]
D. [1, 2, 3, 4, 5, 1, 2, 4, 5]

Answer: A
Explanation: In the given code, a and b are both references to the same list object. When the remove(3) method is called on a, it removes the first occurrence of the value 3 from the list. Since b is a reference to the same list, the change made to a is reflected in b as well, and the output is [1, 2, 4, 5].

Question: 304

What is the purpose of the __delitem__() method in a Python class?

A. To enable deletion of individual elements of the object
B. To define the initial state of the object
C. To specify the default behavior when the object is printed
D. To enable the object to be used in mathematical operations

Answer: A

Explanation: The __delitem__() method in a Python class is used to enable deletion of individual elements of the object. This method is called when you attempt to delete an element of the object using the del keyword, like del obj[index]. By implementing this method, you can define custom behavior for how the object should respond to item deletion operations.

Question: 305
What is the output of the following code snippet?

```
def my_func(x, y):
return round(x / y)

print(my_func(10, 3))
```

A. 3
B. 3.0
C. 3.33

D. 4

Answer: D

Explanation: The my_func takes two parameters x and y and returns the result of x / y rounded to the nearest integer. When my_func(10, 3) is called, it performs the division 10 / 3, which results in 3.3333, and then rounds it to the nearest integer, which is 4.

Question: 306
What is the output of the following code?

```
a = [1, 2, 3, 4, 5]
b = a
a = [10, 20, 30]
print(a, b)
```

A. [10, 20, 30] [10, 20, 30]
B. [10, 20, 30] [1, 2, 3, 4, 5]
C. [1, 2, 3, 4, 5] [10, 20, 30]
D. [1, 2, 3, 4, 5] [1, 2, 3, 4, 5]

Answer: B
Explanation: In the given code, a and b are initially assigned the same list object. However, when a is reassigned to a new list [10, 20, 30], the reference to the original list is lost, and b still points to the original list [1, 2, 3, 4, 5].

Question: 307
What is the output of the following code snippet?

```
def my_func(x, y):
return len(str(x * y))

print(my_func(12, 34))
```

A. 4
B. 5
C. 6
D. 7

Answer: C

Explanation: The my_func takes two parameters x and y, multiplies them, converts the result to a string, and then returns the length of the string. When my_func(12, 34) is called, the result of 12 * 34 is 408, which has a string length of 3. Therefore, the output is 6.

Question: 308
What is the output of the following code?

```
try:
x = 1 / 0
except ZeroDivisionError:
print("ZeroDivisionError occurred")
finally:
print("Finally block executed")
```

A. ZeroDivisionError occurred
Finally block executed
B. Finally block executed
C. ZeroDivisionError occurred
D. TypeError: unsupported operand type(s) for /: 'int' and 'int'

Answer: A
Explanation: The code attempts to divide 1 by 0, which raises a ZeroDivisionError. This error is caught in the except block, and the message "ZeroDivisionError occurred" is printed. Regardless of whether an exception is raised or not, the finally block is always executed, and the message "Finally block executed" is printed.

Question: 309
Which of the following statements about the __new__ method in a Python class is true?

A. It is used to define the behavior of the type() function when used with the class.
B. It is used to define the behavior of the isinstance() function when used with an instance of the class.
C. It is used to define the behavior of the class statement when creating a new class.
D. It is used to define the behavior of the object() function when creating a new instance of the class.

Answer: C
Explanation: The __new__ method in a Python class is used to define the behavior of the class statement when creating a new class, allowing you to customize the creation of the class itself.

Question: 310
What is the output of the following code?

```
class A:
def __init__(self, x):
self.x = x

def method(self):
print("A's method")

class B(A):
def __init__(self, x, y):
A.__init__(self, x)
self.y = y
```

obj = B(1, 2)
print(obj.x, obj.y)
A. 1 2
B. 2 1
C. AttributeError: 'B' object has no attribute 'x'
D. TypeError: init() missing 1 required positional argument: 'y'

Answer: A
Explanation: The B class inherits from the A class and adds the y attribute in its __init__ method. When the obj instance of B is created, the __init__ method of the A class is called with the x argument, and the y argument is assigned to the y attribute of the B class. Therefore, the output is 1 2.

Question: 311
What is the output of the following code?

class A:
def __init__(self):
self.x = 1

class B(A):
def __init__(self):
super().__init__()
self.x = 2

a = A()
b = B()
print(a.x, b.x)
A. 1 1
B. 1 2
C. 2 2
D. An error will be raised

Answer: B
Explanation: In the given code, the A class has an __init__ method that initializes the x attribute to 1. The B class inherits from A and also has an __init__ method that calls the __init__ method of the parent class (A) using super().__init__(), and then sets the x attribute to 2. When instances of A and B are created and their x attributes are printed, the output is 1 2, as the x attribute of the B instance is overwritten by the assignment in the B class's __init__ method.

Question: 312
What is the output of the following code snippet?

```
def my_func(x, y):
return x ** y

print(my_func(2, 3))
```

A. 6
B. 8
C. 9
D. 16

Answer: D

Explanation: The my_func takes two parameters x and y and returns the result of x ** y, which is the exponentiation operation (raising x to the power of y). When my_func(2, 3) is called, it returns the result 2 ** 3 = 8.

Question: 313
What is the output of the following code?

```
def foo(x, y=1, *args, z=2, **kwargs):
print(x, y, args, z, kwargs)
```

foo(0, 1, 2, 3, 4, z=5, a=6, b=7)
A. 0 1 (2, 3, 4) 5 {'a': 6, 'b': 7}
B. 0 1 (2, 3, 4, z=5) {'a': 6, 'b': 7}
C. 0 1 (2, 3, 4) 2 {'z': 5, 'a': 6, 'b': 7}
D. 0 1 (2, 3, 4, 5) {'a': 6, 'b': 7}

Answer: A
Explanation: In the given function signature, x is the first positional argument, y is the second positional argument with a default value of 1, *args collects all the remaining positional arguments into a tuple, z is a keyword-only argument with a default value of 2, and **kwargs collects all the remaining keyword arguments into a dictionary. When the function is called, the arguments are mapped to the corresponding parameters, and the values are printed as specified.

Question: 314
What is the output of the following code?

def func(a, b=1, *args, c=2, **kwargs):
print(a, b, args, c, kwargs)

func(5, 6, 7, 8, c=9, d=10, e=11)
A. 5 6 (7, 8) 9 {'d': 10, 'e': 11}
B. 5 6 (7, 8) 2 {'c': 9, 'd': 10, 'e': 11}
C. 5 1 (7, 8) 9 {'c': 9, 'd': 10, 'e': 11}
D. 5 6 (7, 8) 2 {'d': 10, 'e': 11}

Answer: A
Explanation: The function func() takes the following parameters:

a: a required positional argument
b: an optional positional argument with a default value of 1

*args: a tuple of any additional positional arguments
c: an optional keyword argument with a default value of 2
**kwargs: a dictionary of any additional keyword arguments
When func(5, 6, 7, 8, c=9, d=10, e=11) is called, the arguments are mapped as follows:

a is 5
b is 6
args is the tuple (7, 8)
c is 9 (overriding the default value of 2)
kwargs is the dictionary {'d': 10, 'e': 11}
Therefore, the output is 5 6 (7, 8) 9 {'d': 10, 'e': 11}.

Question: 315
What is the output of the following code?

```
class A:
def __init__(self):
self.x = 1

def __repr__(self):
return f"A(x={self.x})"

a = A()
print(a)
```

A. A(x=1)
B.
C. A
D. 1

Answer: A
Explanation:
The __repr__ method in the A class returns a string representation of the object,

which is used when the object is printed. When print(a) is called, it calls the __repr__ method of the A class, which returns "A(x=1)".

Question: 316
What is the purpose of the __iter__ and __next__ methods in a Python class?

A. To define the behavior of the for loop when iterating over the object.
B. To define the behavior of the in operator when used with the object.
C. To define the behavior of the len() function when used with the object.
D. To define the behavior of the next() function when used with the object.

Answer: A
Explanation: The __iter__ and __next__ methods in a Python class are used to define the behavior of the for loop when iterating over the object, allowing it to be used as an iterator.

Question: 317
What is the output of the following code?

```
def func(x, y):
return x + y

print(func(2, 3) * func(3, 4))
```

A. 25
B. 49
C. 70
D. 77

Answer: B
Explanation: The func(2, 3) call returns 5, and the func(3, 4) call returns 7. The expression func(2, 3) * func(3, 4) then evaluates to 5 * 7 = 35.

Question: 318

What is the purpose of the init.py file in a Python package?

A. It is used to define the package's entry point.
B. It is used to specify the package's dependencies.
C. It is used to initialize the package's global variables.
D. It is used to define the package's modules and subpackages.

Answer: D

Explanation: The init.py file in a Python package serves the purpose of defining the package's modules and subpackages. When a package is imported, the init.py file is executed, and it can be used to perform various initialization tasks, such as setting up the package structure, importing necessary modules, or defining package-level functions and variables.

The other options are incorrect:

A. The entry point of a Python package is typically defined in the setup.py file, not the init.py file.
B. Package dependencies are usually specified in the setup.py file or in a requirements.txt file, not in the init.py file.
C. The init.py file can be used to initialize package-level variables, but this is not its primary purpose.

Question: 319
What is the output of the following code?

```
try:
x = 1 / 0
except ZeroDivisionError:
print("ZeroDivisionError caught")
else:
```

```
print("No exception occurred")
finally:
print("Executing the finally block")
```
A. ZeroDivisionError caught
Executing the finally block
B. ZeroDivisionError caught
No exception occurred
Executing the finally block
C. No exception occurred
Executing the finally block
D. ZeroDivisionError caught

Answer: A
Explanation: The try-except-else-finally block is executed as follows:

The try block attempts to divide 1 by 0, which raises a ZeroDivisionError.
The except block catches the ZeroDivisionError and prints "ZeroDivisionError caught".
The else block is skipped because an exception occurred.
The finally block is executed, printing "Executing the finally block".

Question: 320

What is the output of the following code?

```
a = [1, 2, 3, 4, 5]
b = a[1:4]
c = a[:4]
d = a[:]
print(b, c, d)
```
A. [2, 3, 4] [1, 2, 3, 4] [1, 2, 3, 4, 5]
B. [2, 3, 4] [1, 2, 3, 4] [1, 2, 3, 4, 5]
C. [2, 3, 4] [1, 2, 3, 4] [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
D. [2, 3, 4] [1, 2, 3, 4] [1, 2, 3, 4, 5, 1, 2, 3, 4]

Answer: A
Explanation:

b = a[1:4] creates a new list containing the elements at indices 1, 2, and 3 (2, 3, 4) from the original list a.
c = a[:4] creates a new list containing the elements at indices 0, 1, 2, and 3 (1, 2, 3, 4) from the original list a.
d = a[:] creates a new list that is a copy of the original list a.

Question: 321

What is the output of the following code?

```
class A:
def __init__(self):
self.x = 1
self.y = 2

class B(A):
def __init__(self):
super().__init__()
self.z = 3

b = B()
print(b.x, b.y, b.z)
```

A. 1 2 3
B. 2 3 1
C. AttributeError
D. 1 2

Answer: A
Explanation: The B class inherits from the A class, so it has access to the x and y attributes defined in the A class. In the __init__ method of the B class, super().__init__() is called, which initializes the x and y attributes. The B class also defines its own z attribute, which is then printed along with x and y.

Question: 322
What is the output of the following code?

```
def func(a, b):
try:
c = a / b
print(c)
except ZeroDivisionError:
print("Error: Division by zero")
else:
print("Division successful")

func(10, 2)
func(10, 0)
```

A. 5.0, Error: Division by zero
B. 5.0, Division successful, Error: Division by zero
C. Division successful, Error: Division by zero
D. Error: Division by zero, Division successful

Answer: A
Explanation: The first call to func(10, 2) divides 10 by 2, which is successful, so the output is "5.0" followed by "Division successful". The second call to func(10, 0) divides 10 by 0, which raises a ZeroDivisionError, so the output is "Error: Division by zero".

KILLEXAMS.COM

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including MCQs, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.

Exam Questions:

Killexams.com provides exam questions that are experienced in test centers. These questions are updated regularly to ensure they are up-to-date and relevant to the latest exam syllabus. By studying these questions, candidates can familiarize themselves with the content and format of the real exam.

Exam MCQs:

Killexams.com offers exam MCQs in PDF format. These questions contain a comprehensive collection of questions and answers that cover the exam topics. By using these MCQs, candidate can enhance their knowledge and improve their chances of success in the certification exam.

Practice Test:

Killexams.com provides practice test through their desktop test engine and online test engine. These practice tests simulate the real exam environment and help candidates assess their readiness for the actual exam. The practice test cover a wide range of questions and enable candidates to identify their strengths and weaknesses.

Guaranteed Success:

Killexams.com offers a success guarantee with the exam MCQs. Killexams claim that by using this materials, candidates will pass their exams on the first attempt or they will get refund for the purchase price. This guarantee provides assurance and confidence to individuals preparing for certification exam.

Updated Contents:

Killexams.com regularly updates its question bank of MCQs to ensure that they are current and reflect the latest changes in the exam syllabus. This helps candidates stay up-to-date with the exam content and increases their chances of success.