

DUMPSBOSS.

**PCEP - Certified Entry-Level Python
Programmer**

Python Institute PCEP-30-02

Version Demo

Total Demo Questions: 6

Total Premium Questions: 68

Buy Premium PDF

<https://dumpsboss.co>

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Computer Programming and Python Fundamentals	14
Topic 2, Control Flow – Loops and Conditional Blocks	14
Topic 3, Data Collections – Tuples, Dictionaries, Lists, and Strings	18
Topic 4, Functions	10
Topic 5, Exceptions	8
Topic 6, Mix Questions	4
Total	68

QUESTION NO: 1 - (SIMULATION)

Assuming that the `phone_dir` dictionary contains `namenumbers` pairs, arrange the code boxes to create a valid line of code which retrieves Martin Eden's phone number, and assigns it to the `number` variable.

ANSWER: See the explanation for the answer

Explanation:

Arrange the code to form:

```
number = phone_dir["Martin Eden"]
```

This retrieves the value associated with the `"Martin Eden"` key from the `phone_dir` dictionary and assigns that phone number to `number`.

QUESTION NO: 2

Assuming that the following assignment has been successfully executed:

```
letters = ["a", "b", "c"]
```

Which expressions execute without raising an exception? (Select two answers.)

- A. `letters.index("b")`
- B. `letters.remove("d")`
- C. `letters[3]`
- D. `letters.pop(2)`

ANSWER: A D

Explanation:

`letters.index("b")` is valid because `"b"` occurs in the list, and it returns its index. `letters.pop(2)` is also valid because index 2 identifies the last element of this three-item list.

`letters.remove("d")` raises `ValueError` because `"d"` is not present. `letters[3]` raises `IndexError` because valid indexes are 0, 1, and 2.

QUESTION NO: 3

What is the expected output of the following code?

```
records = {"a": 1}
records["b"] = records["a"] + 2
print(records.get("c", records["b"]))
```

- A. 1
- B. 3
- C. None
- D. The code raises an unhandled exception.

ANSWER: B

Explanation:

The assignment creates the key "b" with the value 3. The dictionary has no key named "c", so `get()` returns its supplied default value, `records["b"]`. Consequently, the code prints 3.

QUESTION NO: 4 - (DRAG DROP)

Drag and drop the code boxes in order to build a program which prints Unavailable to the screen.

(Note: one code box will not be used.)

pass

except KeyError:

except:

```
prices = { "pizza": 3.99 }  
try:  
    charge = prices["calzone"]  
    print("Charged")  
  
    print("Unavailable")  
  
    print("Out of bounds")
```

ANSWER:

Answer:

pass

except KeyError:

except:

```
prices = { "pizza": 3.99 }  
try:  
    charge = prices["calzone"]  
    print("Charged")  
    except KeyError:  
        print("Unavailable")  
    except:  
        print("Out of bounds")
```

Answer:

Explanation:

The completed program uses exception handling to deal with a missing dictionary key. The dictionary assigned to `prices` contains one item, whose key is "pizza". The statement `charge = prices["calzone"]` performs dictionary

subscription with a key that is not present. Python raises `KeyError` for this situation, as described in the [Python documentation for `KeyError`](#).

Putting `except KeyError:` immediately after the `try` suite catches that exact exception. Control then continues in that handler's indented suite, which contains `print("Unavailable")`. Consequently, the requested text is printed to the screen. The statement that would print "Charged" is located after the unsuccessful lookup in the `try` suite, so normal flow never reaches it when the missing-key exception occurs.

The final handler header is `except:`, positioned after the specific `KeyError` handler. It establishes a catch-all handler for an exception not handled by the preceding specific handler, and its suite prints "Out of bounds". This ordering is syntactically valid and follows the normal pattern of placing a specific exception handler before a general handler. Python's [exception-handling tutorial](#) explains that an `except` clause handles an exception raised in its associated `try` suite, and that handlers are examined in order. Since this execution raises `KeyError`, the specific handler is the one selected, producing `Unavailable`.

QUESTION NO: 5 - (SIMULATION)

Arrange the code boxes in the correct positions to form a conditional instruction which guarantees that a certain statement is executed when the speed variable is less than 50.0.

ANSWER: See the explanation for the answer

Explanation:

Arrange the blocks to create this conditional instruction:

```
if speed < 50.0:
    print("The speed is low.")
```

The `if` condition is true when `speed` is less than 50.0, so the indented statement is executed only in that case.

QUESTION NO: 6

What is the expected result of running the following code?

- A. The code prints 1 .
- B. The code prints 2
- C. The code raises an unhandled exception.
- D. The code prints 0

ANSWER: C

Explanation:

The code raises an unhandled exception. Assuming the intended code is `the_list = [1, 2, 3, 4, 5]` followed by `print(the_list.index(6))`, the `index()` list method searches for the first occurrence of its argument and returns that element's zero-based position when it is found. Here, the list contains only the values 1 through 5, so there is no occurrence of the value 6 to return. Python therefore raises `ValueError`, specifically indicating that 6 is not in list. Because there is no `try/except` statement surrounding the call, the exception is unhandled and execution stops while displaying a traceback. This behavior follows the definition of `list.index(x)` in the Python standard library: it returns the index of the first matching item and raises `ValueError` when no item matches. See the [Python documentation for `list.index\(\)`](#) and the [documentation for `ValueError`](#).