Python Interview Questions

Want to become an expert in cracking Python interview questions?

Start with practicing the questions below. Whether a question involves multiple choice or live coding, we will give you hints as you go and tell you if your answers are correct or incorrect.

After that, take our timed public Python Interview Questions Test (/tests/start-challenge?generatorUrl=python-online-test&difficultyHard=False&backUrlSkill=9).

To use our service for testing candidates, buy a pack of candidates (/pricing).

1. File Owners

PYTHON () LANGUAGE () PUBLIC ()

Implement a *group_by_owners* function that:

- Accepts a dictionary containing the file owner name for each file name.
- Returns a dictionary containing a list of file names for each owner name, in any order.

For example, for dictionary *{'Input.txt': 'Randy', 'Code.py': 'Stan', 'Output.txt': 'Randy'}* the *group_by_owners* function should return *{'Randy': ['Input.txt', 'Output.txt'], 'Stan': ['Code.py']}*.

Difficulty	Easy ()
Time	10min

Python 3.6.5 Copy to IDE Show starting code (/questions/original/11846)

```
class FileOwners:

    @staticmethod
    def group_by_owners(files):
        # vinay26k.github.io
        owner_ = {}
        for file, owner in files.items():
            if owner in owner_:
                owner_[owner].append(file)
            else:
                owner_[owner]=[file]
        return owner_

files = {
    'Input.txt': 'Randy',
    'Code.py': 'Stan',
    'Output.txt': 'Randy'
}
print(FileOwners.group_by_owners(files))
```

Run Show Hint

Hints Output Tests: 3 pass / 0 fail

```
Run OK


{'Randy': ['Input.txt', 'Output.txt'], 'Stan': ['Code.py']}
```

```
{ Randy : [ input.txt , output.txt ], Stan : [ code.py ]}
```

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 11846)
Your score is 100%, perfect!

2. Palindrome

PYTHON () STRINGS () PUBLIC ()

A palindrome is a word that reads the same backward or forward.

Write a function that checks if a given word is a palindrome. Character case should be ignored.

For example, *is_palindrome("Deleveled")* should return *True* as character case should be ignored, resulting in "deleveled", which is a palindrome since it reads the same backward and forward.

Difficulty	Easy ()
Time	10min

Python 3.6.5 Copy to IDE Show starting code (/questions/original/7962)

```
class Palindrome:

    @staticmethod
    def is_palindrome(word):
        # vinay26k.github.io
        return word.lower()==word.lower()[::-1]

print(Palindrome.is_palindrome('Deleveled'))
```

Run Show Hint

Hints Output Tests: 3 pass / 0 fail

```
Run OK


True
```

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 7962)
Your score is 100%, perfect!

3. Playlist

PYTHON () **ALGORITHMIC THINKING** () **SEARCHING** () **PUBLIC** () **NEW** ()

A playlist is considered a repeating playlist if any of the songs contain a reference to a previous song in the playlist. Otherwise, the playlist will end with the last song which points to *None*.

Implement a function *is_repeating_playlist* that returns true if a playlist is repeating or false if it is not.

For example, the following code prints "True" as both songs point to each other.

```
first = Song("Hello")
second = Song("Eye of the tiger")

first.next_song(second);
second.next_song(first);

print(first.is_repeating_playlist())
```

Difficulty	Easy ()
Time	15min

Python 3.6.5

Copy to IDE   Show starting code (/questions/original/17253)

```
    def is_repeating_playlist(self):
        """
        :returns: (bool) True if the playlist is repeating, False if not.
        """
        # vinay26k.github.io
        songs_in_playlist = set()
        current_song = self
        while(current_song):
            if current_song.name in songs_in_playlist: # if we already saw th
                return True
            songs_in_playlist.add(current_song.name)
            current_song = current_song.next
        return False

first = Song("Hello")
second = Song("Eye of the tiger")

first.next_song(second);
second.next_song(first);

print(first.is_repeating_playlist())
```

Run

Hints   Output   Tests: 4 pass / 0 fail

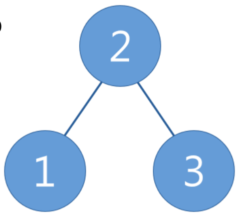
Hint 1: A data structure can be used to identify if a song appears twice in a playlist.

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 17253)
Your score is 100%, perfect!

4. Binary Search Tree

PYTHON () ALGORITHMIC THINKING () DATA STRUCTURES () PUBLIC () NEW ()

Binary search tree (BST) is a binary tree where the value of each node is larger or equal to the values in all the nodes in that node's left subtree and is smaller than the values in all the nodes in that node's right subtree.



Write a function that checks if a given binary search tree contains a given value.

For example, for the following tree:

- n1 (Value: 1, Left: null, Right: null)
- n2 (Value: 2, Left: n1, Right: n3)
- n3 (Value: 3, Left: null, Right: null)

Call to *contains(n2, 3)* should return *True* since a tree with root at n2 contains number 3.

Difficulty	Easy ()
Time	20min

Python 3.6.5 Copy to IDE Show starting code (/questions/original/14288)

```
    @staticmethod
    def contains(root, value):
        # vinay26k.github.io
        if root is None:
            return False
        if root.value == value:
            return True
        elif root.value < value:
            if root.right:
                return BinarySearchTree.contains(root.right,value)
        elif root.value > value:
            if root.left:
                return BinarySearchTree.contains(root.left, value)
        return False

n1 = BinarySearchTree.Node(value=1, left=None, right=None)
n3 = BinarySearchTree.Node(value=3, left=None, right=None)
n2 = BinarySearchTree.Node(value=2, left=n1, right=n3)

```

Run Show Hint

Hints | Output | Tests: 3 pass / 0 fail

```
Run OK

True
```

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 14288)
Your score is 100%, perfect!

5. Two Sum

PYTHON () **ALGORITHMIC THINKING** () **SEARCHING** () **PUBLIC** () **NEW** ()

Write a function that, when passed a list and a target sum, returns two distinct zero-based indices of any two of the numbers, whose sum is equal to the target sum. If there are no two numbers, the function should return *None*.

For example, *find_two_sum([3, 1, 5, 7, 5, 9], 10)* should return a single *tuple* containing any of the following pairs of indices:

- 0 and 3 (or 3 and 0) as 3 + 7 = 10
- 1 and 5 (or 5 and 1) as 1 + 9 = 10
- 2 and 4 (or 4 and 2) as 5 + 5 = 10

Difficulty	Easy ()
Time	30min

Python 3.6.5 Copy to IDE Show starting code (/questions/original/16305)

```
class TwoSum:

    @staticmethod
    def find_two_sum(numbers, target_sum):
        """
        :param numbers: (list of ints) The list of numbers.
        :param target_sum: (int) The required target sum.
        :returns: (a tuple of 2 ints) The indices of the two elements whose su
        """
        # vinay26k.github.io
        if len(numbers) <= 1:
            return False
        buff_dict = {}
        for i in range(len(numbers)):
            if numbers[i] in buff_dict:
                return (buff_dict[numbers[i]], i)
            else:
                buff_dict[target_sum - numbers[i]] = i

```

Run Show Hint

Hints Output Tests: 4 pass / 0 fail

```
Run OK

(0, 3)
```

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 16305)
Your score is 100%, perfect!

6. League Table

PYTHON () COLLECTIONS () SORTING () PUBLIC () NEW ()

The *LeagueTable* class tracks the score of each player in a league. After each game, the player records their score with the *record_result* function.

The player's rank in the league is calculated using the following logic:

1. The player with the highest score is ranked first (rank 1). The player with the lowest score is ranked last.
2. If two players are tied on score, then the player who has played the fewest games is ranked higher.
3. If two players are tied on score and number of games played, then the player who was first in the list of players is ranked higher.

Implement the *player_rank* function that returns the player at the given rank.

For example:

```
table = LeagueTable(['Mike', 'Chris', 'Arnold'])
table.record_result('Mike', 2)
table.record_result('Mike', 3)
table.record_result('Arnold', 5)
table.record_result('Chris', 5)
print(table.player_rank(1))
```

All players have the same score. However, Arnold and Chris have played fewer games than Mike, and as Chris is before Arnold in the list of players, he is ranked first. Therefore, the code above should display "Chris".

Difficulty	Hard ()
Time	20min

Python 3.6.5 ? Copy to IDE Show starting code (/questions/original/11195)

```
class LeagueTable:
    def __init__(self, players):
        self.standings = OrderedDict([(player, Counter()) for player in playe

    def record_result(self, player, score):
        self.standings[player]['games_played'] += 1
        self.standings[player]['score'] += score

    def player_rank(self, rank):
        # vinay26k.github.io
        ranks = []
        for player in self.standings:
            ranks.append((player, self.standings[player]['games_played'], sel
        return sorted(ranks, key=lambda x: (-x[2], x[1]))[rank-1][0]

table = LeagueTable(['Mike', 'Chris', 'Arnold'])
table.record_result('Mike', 2)
table.record_result('Mike', 3)
table.record_result('Arnold', 5)
table.record_result('Chris', 5)
```

Run Show Hint

Hints Output Tests: 4 pass / 0 fail

```
Run OK

Chris
```

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 11195)
Your score is 100%, perfect!

7. Path

PYTHON () **DATA STRUCTURES** () **STRINGS** () **PUBLIC** ()

Write a function that provides change directory (cd) function for an abstract file system.

Notes:

- Root path is '/'.
- Path separator is '/'.
- Parent directory is addressable as '..'.
- Directory names consist only of English alphabet letters (A-Z and a-z).
- The function should support both relative and absolute paths.
- The function will not be passed any invalid paths.
- Do not use built-in path-related functions.

For example:

```
path = Path('/a/b/c/d')
path.cd('../x')
print(path.current_path)
```

should display '/a/b/c/x'.

Difficulty	Hard ()
Time	30min

Python 3.6.5 Copy to IDE Show starting code (/questions/original/12282)

```
class Path:
    def __init__(self, path):
        self.current_path = path

    def cd(self, new_path):
        # vinay26k.github.io
        go_back_count = new_path.split('/').count('..')
        if go_back_count:
            self.current_path = '/'.join(self.current_path.split('/')[:-go_bac
        else:
            self.current_path += '/'+new_path


path = Path('/a/b/c/d')
path.cd('../x')
print(path.current_path)
```

Run Show Hint

Hints Output Tests: 4 pass / 0 fail

```
Run OK

/a/b/c/x
```

Report an issue (mailto:support@testdome.com?subject=Report question: https://www.testdome.com/d/python-interview-questions/9, QuestionId: 12282)
Your score is 100%, perfect!

If you feel ready, take one of our timed public Python Interview Questions tests: