Dashboard / My courses / CD19411-PPD-2022 / WEEK_06-Strings / WEEK-06_CODING

Started on	Tuesday, 9 April 2024, 4:37 AM
State	Finished
Completed on	Saturday, 13 April 2024, 2:36 PM
Time taken	4 days 9 hours
Marks	4.00/5.00
Grade	**40.00** out of 50.00 (**80**%)
Name	GOWRI NANDA M 2022-CSD-A

Question **1**

Correct

Mark 1.00 out of 1.00

Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

For example:

Input	Result
`break`	`break is a keyword`
`IF`	`IF is not a keyword`

Answer: (penalty regime: 0 %)

```
a=(input())
b=["break","case","continue","default","defer","else","for","func","goto","if","map","range","return",
if a in b:
    print(a,"is a keyword");
else:
    print(a,"is not a keyword");
```

	Input	Expected	Got	
✓	`break`	`break is a keyword`	`break is a keyword`	✓

	Input	Expected	Got	
✓	`IF`	`IF is not a keyword`	`IF is not a keyword`	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

```
Input:
A man, a plan, a canal: Panama

Output:
1
```

Example 2:

```
Input:
race a car

Output:
0
```

Constraints:

- `s` consists only of printable ASCII characters.

Answer: (penalty regime: 0 %)

```
string= input()
str=""
for i in range(len(string)):
    if(string[i].isalnum()):
        str+=string[i]


if str.lower()==str[::-1].lower():
    print(1)
else:
    print(0)
```

	Input	Expected	Got	
✓	A man, a plan, a canal: Panama	1	1	✓
✓	race a car	0	0	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Verify the given number is cyclic or not.

Input Format

Num1

Num2

Constraints

1<=range<=9999999999

Sample Input 1

12345

45123

Sample Output 1

Yes

Sample Input 2

12345

54123

Sample Output 2

No

Answer: (penalty regime: 0 %)

```python
def is_rotation(num1, num2):
    str_num1 = str(num1)
    str_num2 = str(num2)

    if len(str_num1) != len(str_num2):
        return "No"

    doubled_num1 = str_num1 + str_num1

    if str_num2 in doubled_num1:
        return "Yes"
    else:
        return "No"

try:
    num1 = int(input().strip())
    num2 = int(input().strip())

    # Check if Num2 is a rotation of Num1
    result = is_rotation(num1, num2)

    # Print the result
```

	Input	Expected	Got	
✓	12345 45123	Yes	Yes	✓
✓	12345 54123	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Find if a String2 is substring of String1. If it is, return the index of the first occurrence. else return -1.

Sample Input 1

thistest123string

123

Sample Output 1

8

Answer: (penalty regime: 0 %)

```
str1=str(input())
str2=str(input())
print(str1.find(str2))
```

	Input	Expected	Got	
✓	thistest123string 123	8	8	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Incorrect

Mark 0.00 out of 1.00

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

Constraints:

1 <= s.length <= 10^4

s consists of parentheses only '()[]{}'.

For example:

Input	Result
`()`	`true`
`()[]{}`	`true`
`(]`	`false`

Answer: (penalty regime: 0 %)

```python
def is_valid(s):
    stack = []
    brackets_map = {')': '(', ']': '[', '}': '{'}

    for char in s:
        if char in brackets_map.values():
            stack.append(char)
        elif char in brackets_map:
            if not stack or stack[-1] != brackets_map[char]:
                return False
            stack.pop()

    return not stack

# Test cases
test_cases = [
    ("()", True),
    ("()[]{}", True),
    ("(]", False)
]

for i, (test_input, expected_output) in enumerate(test_cases):
```

	Input	Expected	Got	

✖	`()`	`true`	`True`
			`True`
			`False`
				✖

✖	`()[]{}`	`true`	`True`
			`True`
			`False`
				✖

	Input	Expected	Got	

✖	`(]`	`false`	`True`
			`True`
			`False`
				✖

Some hidden test cases failed, too.

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

◄ Week-06_MCQ

Jump to...

WEEK-06-Extra ►