Skip to main content

CS23336-Introduction to Python Programming

Started on	Friday, 18 October 2024, 12:34 PM
State	Finished
Completed on	Friday, 18 October 2024, 1:11 PM
Time taken	36 mins 4 secs
Marks	10.00/10.00
Grade	**100.00** out of 100.00

Question 1

Correct
Mark 1.00 out of 1.00
Flag question

Question text

A pangram is a sentence where every letter of the English alphabet appears at least once.

Given a string sentence containing only lowercase English letters, return true if sentence is a pangram, or false otherwise.

Example 1:

Input:

thequickbrownfoxjumpsoverthelazydog

Output:

true

Explanation: sentence contains at least one of every letter of the English alphabet.

Example 2:

Input:

arvijayakumar

Output: false

Constraints:

1 <= sentence.length <= 1000

sentence consists of lowercase English letters.

For example:

Test	Result
`print(checkPangram('thequickbrownfoxjumpsoverthelazydog'))`	`true`
`print(checkPangram('arvijayakumar'))`	`false`

Answer:(penalty regime: 0 %)

Reset answer

```python
import string
def checkPangram(s):
    a=set(string.ascii_lowercase)
    b=set(c.lower()for c in s if c.isalpha())
    return 'true' if a<=b else 'false'
```

Feedback

Test	Expected	Got
`print(checkPangram('thequickbrownfoxjumpsoverthelazydog'))`	`true`	`true`
`print(checkPangram('arvijayakumar'))`	`false`	`false`

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 2

Correct
Mark 1.00 out of 1.00
Flag question

Question text

The program must accept **N** series of keystrokes as string values as the input. The character ^ represents undo action to clear the last entered keystroke. The program must print the string typed after applying the undo operations as the output. If there are no characters in the string then print **-1** as the output.

Boundary Condition(s):

1 <= N <= 100
1 <= Length of each string <= 100

Input Format:

The first line contains the integer N.
The next N lines contain a string on each line.

Output Format:

The first N lines contain the string after applying the undo operations.

Example Input/Output 1:

Input:

```
3
Hey ^ goooo^^glee^
lucke^y ^charr^ms
ora^^nge^^^^
```

Output:

```
Hey google
luckycharms
-1
```

Answer:(penalty regime: 0 %)

```
def pk(n,b):
    for j in b:
        result=[]
        for char in j:
```

```python
        if char=='^':
            if result:
                result.pop()
        else:
            result.append(char)
    if result:
        print("".join(result))
    else:
        print(-1)
n=int(input())
b=[input()for i in range(n)]
pk(n,b)
```

Feedback

Input	Expected	Got
3		
Hey ^ goooo^^glee^	Hey google	Hey google
lucke^y ^charr^ms	luckycharms	luckycharms
ora^^nge^^^^	-1	-1

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 3

Correct
Mark 1.00 out of 1.00
Flag question

Question text

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 %)

```python
a=input()
b=""
c=""
for i in a:
    if i.isalnum():
        b+=i.lower()
c=b[::-1]
if c==b:
    print(1)
else:
    print(0)
```

Feedback

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 4

Correct

Mark 1.00 out of 1.00
Flag question

Question text

Given a **non-empty** string s and an abbreviation abbr, return whether the string matches with the given abbreviation.

A string such as "word" contains only the following valid abbreviations:

["word", "1ord", "w1rd", "wo1d", "wor1", "2rd", "w2d", "wo2", "1o1d", "1or1", "w1r1", "1o2", "2r1", "3d", "w3", "4"]

Notice that only the above abbreviations are valid abbreviations of the string "word". Any other string is not a valid abbreviation of "word".

Note:

Assume s contains only lowercase letters and abbr contains only lowercase letters and digits.

Example 1:

Input

internationalization

i12iz4n

Output

true

Explanation

Given **s** = "internationalization", **abbr** = "i12iz4n":

Return true.

Example 2:

Input

apple

a2e

Output

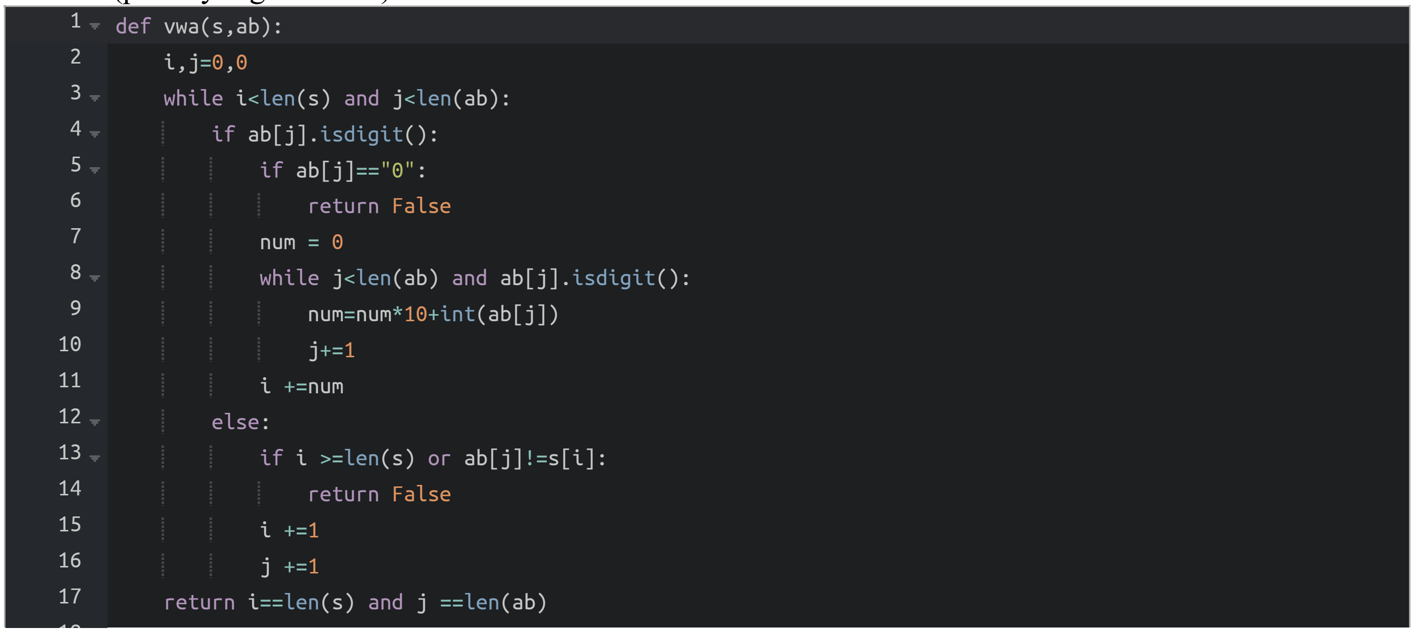
false

Explanation

Given **s** = "apple", **abbr** = "a2e":

Return false.

Answer:(penalty regime: 0 %)



```python
def vwa(s,ab):
    i,j=0,0
    while i<len(s) and j<len(ab):
        if ab[j].isdigit():
            if ab[j]=="0":
                return False
            num = 0
            while j<len(ab) and ab[j].isdigit():
                num=num*10+int(ab[j])
                j+=1
            i +=num
        else:
            if i >=len(s) or ab[j]!=s[i]:
                return False
            i +=1
            j +=1
    return i==len(s) and j ==len(ab)
```

```
s=input()
ab=input()
x=vwa(s,ab)
print('true' if x else 'false')
```

Feedback

Input	Expected	Got
internationalization i12iz4n	true	true
apple a2e	false	false

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 5

Correct
Mark 1.00 out of 1.00
Flag question

Question text

Assume that the given string has enough memory.

Don't use any extra space(IN-PLACE)

Sample Input 1

a2b4c6

Sample Output 1

aabbbbcccccc
Answer:(penalty regime: 0 %)

```
def ds(a):
    result=[]
    i=0
```

```python
    while i<len(a):
        char=a[i]
        count=""
        i+=1
        while i<len(a) and a[i].isdigit():
            count +=a[i]
            i+=1
        ct=int(count)
        result.append(char*ct)
    return''.join(result)
a=input()
x=ds(a)
print(x)
```

Feedback

Input	Expected	Got
`a2b4c6`	`aabbbbcccccc`	`aabbbbcccccc`
`a12b3d4`	`aaaaaaaaaaaabbbdddd`	`aaaaaaaaaaaabbbdddd`

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 6

Correct
Mark 1.00 out of 1.00
Flag question

Question text

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

Feedback

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

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 7

Correct
Mark 1.00 out of 1.00
Flag question

Question text

Given a string S which is of the format USERNAME@DOMAIN.EXTENSION, the program must print the EXTENSION, DOMAIN, USERNAME in the reverse order.

Input Format:

The first line contains S.

Output Format:

The first line contains EXTENSION.
The second line contains DOMAIN.
The third line contains USERNAME.

Boundary Condition:

1 <= Length of S <= 100

Example Input/Output 1:

Input:

abcd@gmail.com

Output:

com
gmail
abcd

For example:

Input	Result

`arvijayakumar@rajalakshmi.edu.in`	`edu.in`
	`rajalakshmi`
	`arvijayakumar`

Answer:(penalty regime: 0 %)

```
a=input()
b=a.find('@')
c=a.find('.')
print(a[c+1:len(a)])
print(a[b+1:c])
print(a[0:b])
```

Feedback

Input	Expected	Got

`abcd@gmail.com`	`com`
	`gmail`
	`abcd`
		`com`
		`gmail`
		`abcd`

`arvijayakumar@rajalakshmi.edu.in`	`edu.in`
	`rajalakshmi`
	`arvijayakumar`
		`edu.in`
		`rajalakshmi`
		`arvijayakumar`

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 8

Correct
Mark 1.00 out of 1.00
Flag question

Question text

Write a Python program to get one string and reverses a string. The input string is given as an array of characters `char[]`.

You may assume all the characters consist of printable ascii characters.

Example 1:

```
Input:
hello
Output:
olleh
```

Example 2:

```
Input:
Hannah
Output:
hannaH
```

Answer:(penalty regime: 0 %)

```python
a=input()
print(a[::-1])
```

Feedback

Input	Expected	Got
hello	olleh	olleh
Hannah	hannaH	hannaH

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 9

Correct
Mark 1.00 out of 1.00
Flag question

Question text

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 %)

```
a=input()
b=input()
print(a.find(b))
```

Feedback

Input	Expected	Got
thistest123string 123	8	8

Passed all tests!
Correct
Marks for this submission: 1.00/1.00.

Question 10

Correct
Mark 1.00 out of 1.00
Flag question

Question text

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:

Test	Result
`print(ValidParenthesis("()"))`	`true`
`print(ValidParenthesis("()[]{}"))`	`true`
`print(ValidParenthesis("(]"))`	`false`

Answer:(penalty regime: 0 %)
Reset answer

```python
def ValidParenthesis(s):
    stack=[]
    mp={')':'(','}':'{',']':'['}
    for char in s:
        if char in mp.values():
            stack.append(char)
        elif char in mp:
            if not stack or stack[-1]!=mp[char]:
                return 'false'
            stack.pop()
    return 'true' if not stack else 'false'
```

Feedback

Test	Expected	Got
`print(ValidParenthesis("()"))`	`true`	`true`
`print(ValidParenthesis("()[]{}"))`	`true`	`true`
`print(ValidParenthesis("(]"))`	`false`	`false`

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.
Finish review