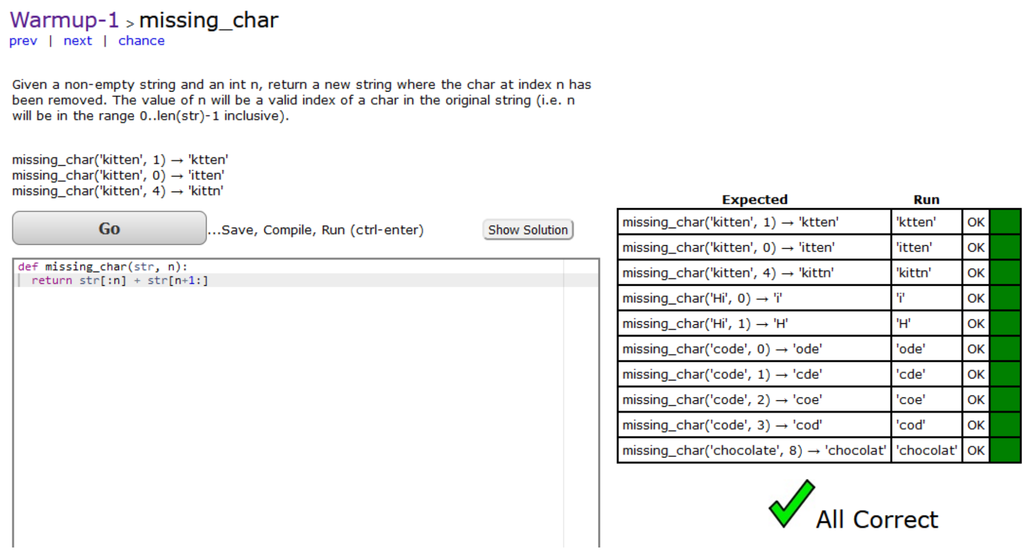
Coding Bat – First Call



Warmup-1 > missing_char

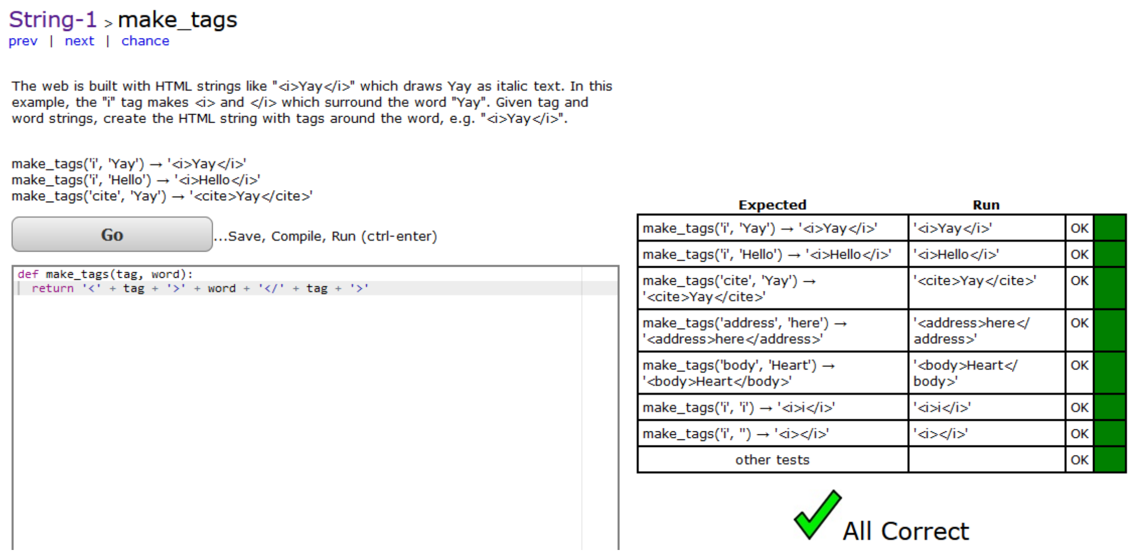
Given a non-empty string and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0..len(str)-1 inclusive).

missing_char('kitten', 1) → 'ktten'
missing_char('kitten', 0) → 'itten'
missing_char('kitten', 4) → 'kittn'

```
def missing_char(str, n):
  return str[:n] + str[n+1:]
```

Expected	Run		
missing_char('kitten', 1) → 'ktten'	'ktten'	OK	🟩
missing_char('kitten', 0) → 'itten'	'itten'	OK	🟩
missing_char('kitten', 4) → 'kittn'	'kittn'	OK	🟩
missing_char('Hi', 0) → 'i'	'i'	OK	🟩
missing_char('Hi', 1) → 'H'	'H'	OK	🟩
missing_char('code', 0) → 'ode'	'ode'	OK	🟩
missing_char('code', 1) → 'cde'	'cde'	OK	🟩
missing_char('code', 2) → 'coe'	'coe'	OK	🟩
missing_char('code', 3) → 'cod'	'cod'	OK	🟩
missing_char('chocolate', 8) → 'chocolat'	'chocolat'	OK	🟩

✓ All Correct



String-1 > make_tags

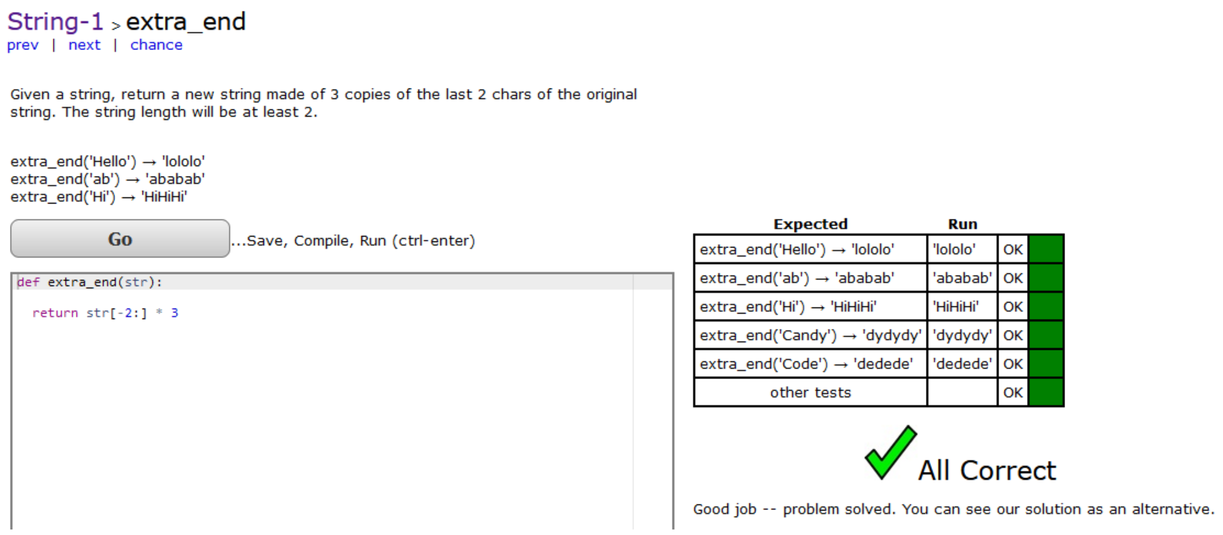
The web is built with HTML strings like "<i>Yay</i>" which draws Yay as italic text. In this example, the "i" tag makes <i> and </i> which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "<i>Yay</i>".

make_tags('i', 'Yay') → '<i>Yay</i>'
make_tags('i', 'Hello') → '<i>Hello</i>'
make_tags('cite', 'Yay') → '<cite>Yay</cite>'

```
def make_tags(tag, word):
  return '<' + tag + '>' + word + '</' + tag + '>'
```

Expected	Run		
make_tags('i', 'Yay') → '<i>Yay</i>'	'<i>Yay</i>'	OK	🟩
make_tags('i', 'Hello') → '<i>Hello</i>'	'<i>Hello</i>'	OK	🟩
make_tags('cite', 'Yay') → '<cite>Yay</cite>'	'<cite>Yay</cite>'	OK	🟩
make_tags('address', 'here') → '<address>here</address>'	'<address>here</address>'	OK	🟩
make_tags('body', 'Heart') → '<body>Heart</body>'	'<body>Heart</body>'	OK	🟩
make_tags('i', 'i') → '<i>i</i>'	'<i>i</i>'	OK	🟩
make_tags('i', '') → '<i></i>'	'<i></i>'	OK	🟩
other tests		OK	🟩

✓ All Correct



String-1 > extra_end

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

extra_end('Hello') → 'lololo'
extra_end('ab') → 'ababab'
extra_end('Hi') → 'HiHiHi'

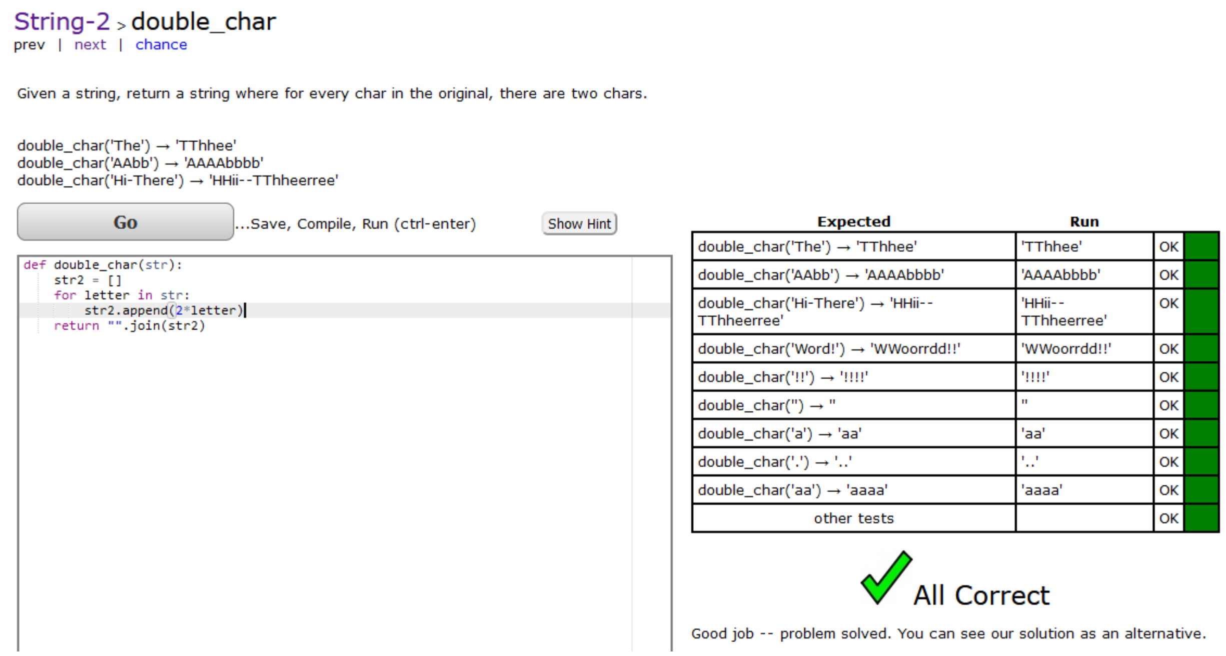
```
def extra_end(str):

  return str[-2:] * 3
```

Expected	Run		
extra_end('Hello') → 'lololo'	'lololo'	OK	🟩
extra_end('ab') → 'ababab'	'ababab'	OK	🟩
extra_end('Hi') → 'HiHiHi'	'HiHiHi'	OK	🟩
extra_end('Candy') → 'dydydy'	'dydydy'	OK	🟩
extra_end('Code') → 'dedede'	'dedede'	OK	🟩
other tests		OK	🟩

✓ All Correct

Good job -- problem solved. You can see our solution as an alternative.



String-2 > double_char

prev | next | chance

Given a string, return a string where for every char in the original, there are two chars.

double_char('The') → 'TThhee'
double_char('AAbb') → 'AAAAbbbb'
double_char('Hi-There') → 'HHii--TThheerree'

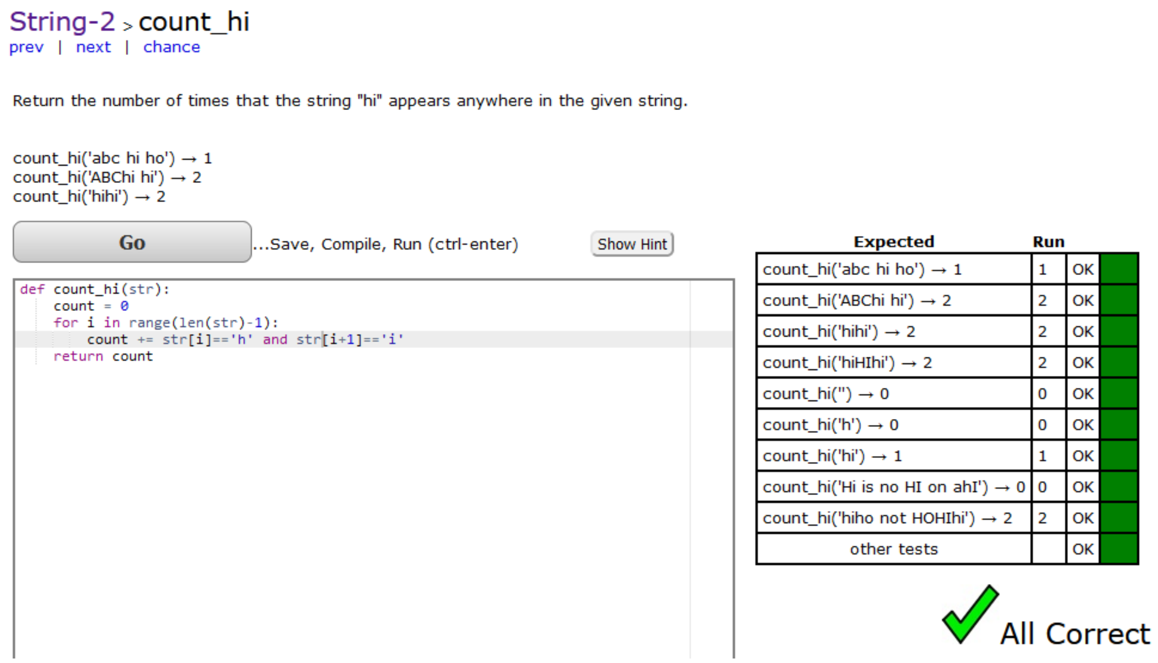
Go ...Save, Compile, Run (ctrl-enter) Show Hint

```python
def double_char(str):
    str2 = []
    for letter in str:
        str2.append(2*letter)
    return "".join(str2)
```

Expected	Run		
double_char('The') → 'TThhee'	'TThhee'	OK	🟩
double_char('AAbb') → 'AAAAbbbb'	'AAAAbbbb'	OK	🟩
double_char('Hi-There') → 'HHii--TThheerree'	'HHii--TThheerree'	OK	🟩
double_char('Word!') → 'WWoorrdd!!'	'WWoorrdd!!'	OK	🟩
double_char('!!') → '!!!!'	'!!!!'	OK	🟩
double_char('') → ''	''	OK	🟩
double_char('a') → 'aa'	'aa'	OK	🟩
double_char('.') → '..'	'..'	OK	🟩
double_char('aa') → 'aaaa'	'aaaa'	OK	🟩
other tests		OK	🟩

✓ All Correct

Good job -- problem solved. You can see our solution as an alternative.



String-2 > count_hi

prev | next | chance

Return the number of times that the string "hi" appears anywhere in the given string.

count_hi('abc hi ho') → 1
count_hi('ABChi hi') → 2
count_hi('hihi') → 2

Go ...Save, Compile, Run (ctrl-enter) Show Hint

```python
def count_hi(str):
    count = 0
    for i in range(len(str)-1):
        count += str[i]=='h' and str[i+1]=='i'
    return count
```

Expected	Run		
count_hi('abc hi ho') → 1	1	OK	🟩
count_hi('ABChi hi') → 2	2	OK	🟩
count_hi('hihi') → 2	2	OK	🟩
count_hi('hiHIhi') → 2	2	OK	🟩
count_hi('') → 0	0	OK	🟩
count_hi('h') → 0	0	OK	🟩
count_hi('hi') → 1	1	OK	🟩
count_hi('Hi is no HI on ahI') → 0	0	OK	🟩
count_hi('hiho not HOHIhi') → 2	2	OK	🟩
other tests		OK	🟩

✓ All Correct