CodingBat code practice

Java | Python

Logic-1 > date_fashion

prev | next | chance

You and your date are trying to get a table at a restaurant. The parameter "you" is the stylishness of your clothes, in the range 0..10, and "date" is the stylishness of your date's clothes. The result getting the table is encoded as an int value with 0=no, 1=maybe, 2=yes. If either of you is very stylish, 8 or more, then the result is 2 (yes). With the exception that if either of you has style of 2 or less, then the result is 0 (no). Otherwise the result is 1 (maybe).

date_fashion(5, 10) → 2
date_fashion(5, 2) → 0
date_fashion(5, 5) → 1

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

```python
def date_fashion(you, date):
  if you <= 2 or date <= 2:
      return 0
  elif you >= 8 or date >= 8:
      return 2
  else:
      return 1
```

Go

Editor font size %: 100

Shorter output ☐

Expected	Run		
date_fashion(5, 10) → 2	2	OK	🟩
date_fashion(5, 2) → 0	0	OK	🟩
date_fashion(5, 5) → 1	1	OK	🟩
date_fashion(3, 3) → 1	1	OK	🟩
date_fashion(10, 2) → 0	0	OK	🟩
date_fashion(2, 9) → 0	0	OK	🟩
date_fashion(9, 9) → 2	2	OK	🟩
date_fashion(10, 5) → 2	2	OK	🟩
date_fashion(2, 2) → 0	0	OK	🟩
date_fashion(3, 7) → 1	1	OK	🟩
date_fashion(2, 7) → 0	0	OK	🟩
date_fashion(6, 2) → 0	0	OK	🟩
other tests		OK	🟩

✓ All Correct

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

See Our Solution

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem

CodingBat code practice

Java | Python

Logic-1 > squirrel_play

prev | next | chance

The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 (inclusive). Unless it is summer, then the upper limit is 100 instead of 90. Given an int temperature and a boolean is_summer, return True if the squirrels play and False otherwise.

squirrel_play(70, False) → True
squirrel_play(95, False) → False
squirrel_play(95, True) → True

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

```
def squirrel_play(temp, is_summer):
  if is_summer:
    return 60 <= temp <= 100
  else:
    return 60 <= temp <= 90
```

Go

Editor font size %: 100

Shorter output

Expected	Run		
squirrel_play(70, False) → True	True	OK	🟩
squirrel_play(95, False) → False	False	OK	🟩
squirrel_play(95, True) → True	True	OK	🟩
squirrel_play(90, False) → True	True	OK	🟩
squirrel_play(90, True) → True	True	OK	🟩
squirrel_play(50, False) → False	False	OK	🟩
squirrel_play(50, True) → False	False	OK	🟩
squirrel_play(100, False) → False	False	OK	🟩
squirrel_play(100, True) → True	True	OK	🟩
squirrel_play(105, True) → False	False	OK	🟩
squirrel_play(59, False) → False	False	OK	🟩
squirrel_play(59, True) → False	False	OK	🟩
squirrel_play(60, False) → True	True	OK	🟩
other tests		OK	🟩

✓ All Correct

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem

Forget It! -- delete my code for this problem

CodingBat code practice

Java | Python

Logic-1 > caught_speeding

prev | next | chance

You are driving a little too fast, and a police officer stops you. Write code to compute the result, encoded as an int value: 0=no ticket, 1=small ticket, 2=big ticket. If speed is 60 or less, the result is 0. If speed is between 61 and 80 inclusive, the result is 1. If speed is 81 or more, the result is 2. Unless it is your birthday -- on that day, your speed can be 5 higher in all cases.

caught_speeding(60, False) → 0
caught_speeding(65, False) → 1
caught_speeding(65, True) → 0

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

```
def caught_speeding(speed, is_birthday):
  if is_birthday:
      speed -= 5
  if speed <= 60:
      return 0
  elif 61 <= speed <= 80:
      return 1
  else:
      return 2
```

Go

Editor font size %: 100

Shorter output ☐

Expected	Run		
caught_speeding(60, False) → 0	0	OK	🟩
caught_speeding(65, False) → 1	1	OK	🟩
caught_speeding(65, True) → 0	0	OK	🟩
caught_speeding(80, False) → 1	1	OK	🟩
caught_speeding(85, False) → 2	2	OK	🟩
caught_speeding(85, True) → 1	1	OK	🟩
caught_speeding(70, False) → 1	1	OK	🟩
caught_speeding(75, False) → 1	1	OK	🟩
caught_speeding(75, True) → 1	1	OK	🟩
caught_speeding(40, False) → 0	0	OK	🟩
caught_speeding(40, True) → 0	0	OK	🟩
caught_speeding(90, False) → 2	2	OK	🟩
other tests		OK	🟩

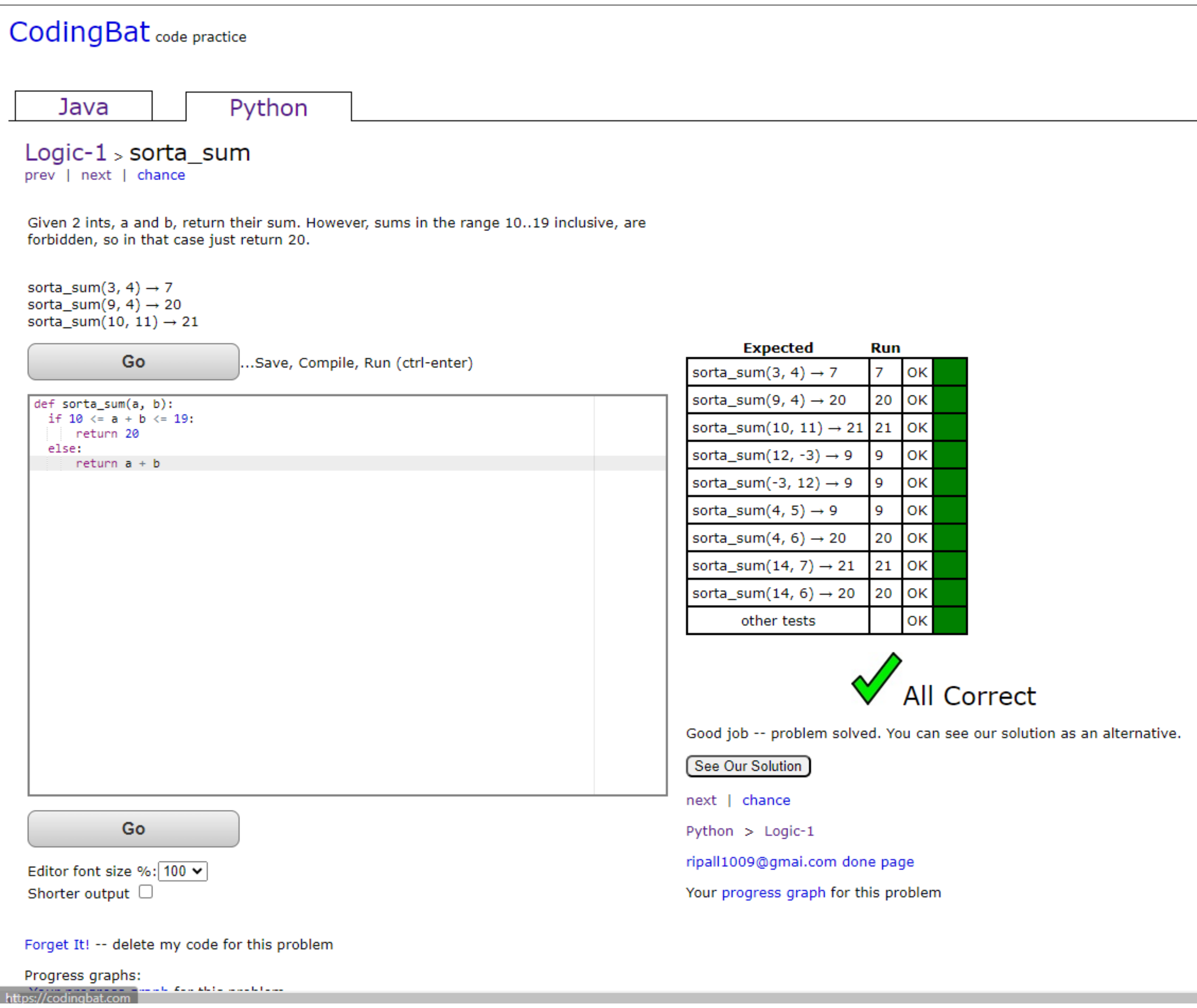
✓ All Correct

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem



CodingBat code practice

Logic-1 > sorta_sum

Given 2 ints, a and b, return their sum. However, sums in the range 10..19 inclusive, are forbidden, so in that case just return 20.

sorta_sum(3, 4) → 7
sorta_sum(9, 4) → 20
sorta_sum(10, 11) → 21

...Save, Compile, Run (ctrl-enter)

```python
def sorta_sum(a, b):
  if 10 <= a + b <= 19:
      return 20
  else:
      return a + b
```

Editor font size %: 100

Shorter output ☐

Expected	Run		
sorta_sum(3, 4) → 7	7	OK	🟩
sorta_sum(9, 4) → 20	20	OK	🟩
sorta_sum(10, 11) → 21	21	OK	🟩
sorta_sum(12, -3) → 9	9	OK	🟩
sorta_sum(-3, 12) → 9	9	OK	🟩
sorta_sum(4, 5) → 9	9	OK	🟩
sorta_sum(4, 6) → 20	20	OK	🟩
sorta_sum(14, 7) → 21	21	OK	🟩
sorta_sum(14, 6) → 20	20	OK	🟩
other tests		OK	🟩

✓ All Correct

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

ripall1009@gmai.com done page

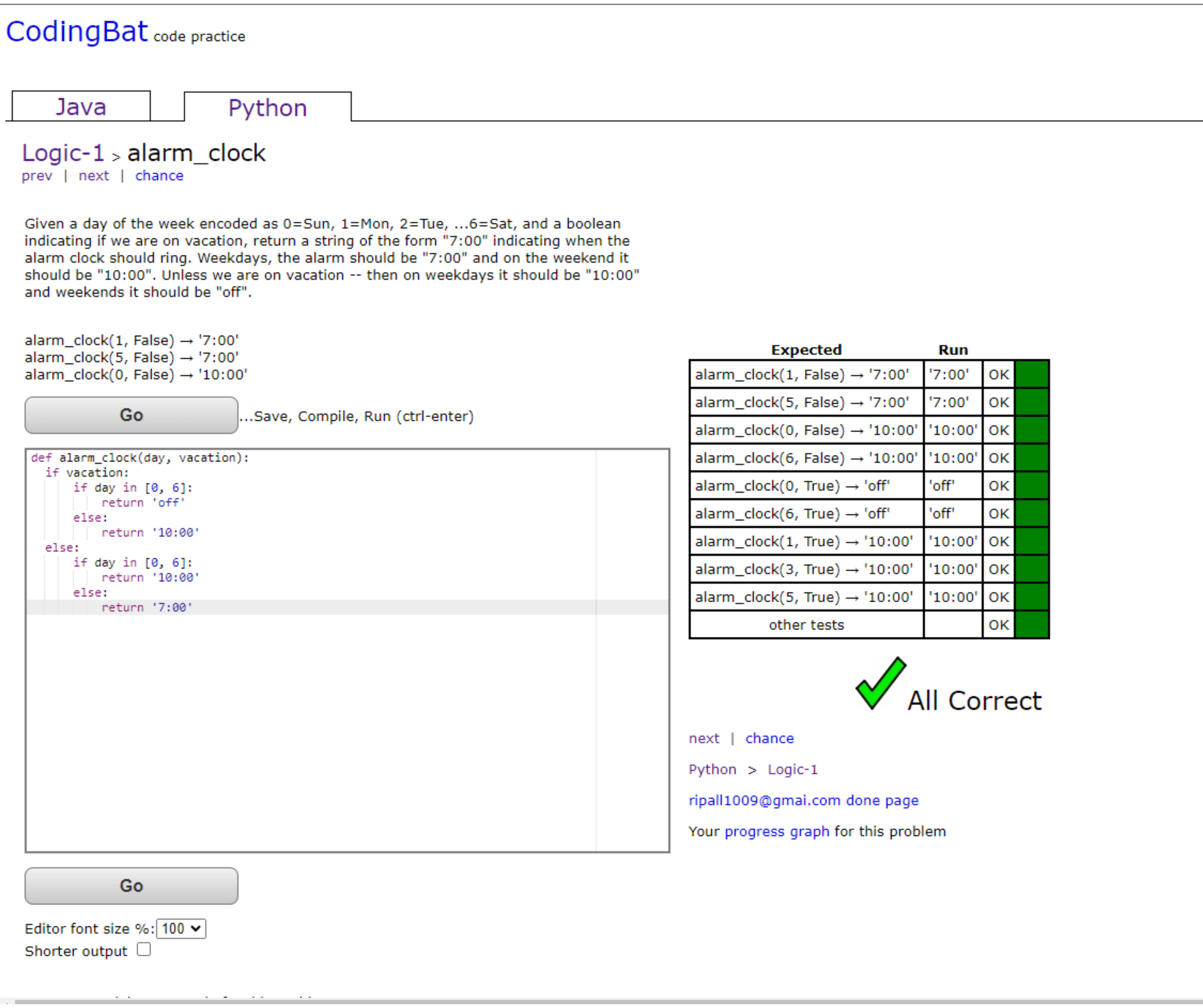
Your progress graph for this problem

Forget It! -- delete my code for this problem

Progress graphs:

Your progress graph for this problem

https://codingbat.com



CodingBat code practice

Java | Python

Logic-1 > alarm_clock

prev | next | chance

Given a day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation, return a string of the form "7:00" indicating when the alarm clock should ring. Weekdays, the alarm should be "7:00" and on the weekend it should be "10:00". Unless we are on vacation -- then on weekdays it should be "10:00" and weekends it should be "off".

alarm_clock(1, False) → '7:00'
alarm_clock(5, False) → '7:00'
alarm_clock(0, False) → '10:00'

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

```
def alarm_clock(day, vacation):
  if vacation:
      if day in [0, 6]:
          return 'off'
      else:
          return '10:00'
  else:
      if day in [0, 6]:
          return '10:00'
      else:
          return '7:00'
```

Go

Editor font size %: 100

Shorter output ☐

Expected	Run		
alarm_clock(1, False) → '7:00'	'7:00'	OK	🟩
alarm_clock(5, False) → '7:00'	'7:00'	OK	🟩
alarm_clock(0, False) → '10:00'	'10:00'	OK	🟩
alarm_clock(6, False) → '10:00'	'10:00'	OK	🟩
alarm_clock(0, True) → 'off'	'off'	OK	🟩
alarm_clock(6, True) → 'off'	'off'	OK	🟩
alarm_clock(1, True) → '10:00'	'10:00'	OK	🟩
alarm_clock(3, True) → '10:00'	'10:00'	OK	🟩
alarm_clock(5, True) → '10:00'	'10:00'	OK	🟩
other tests		OK	🟩

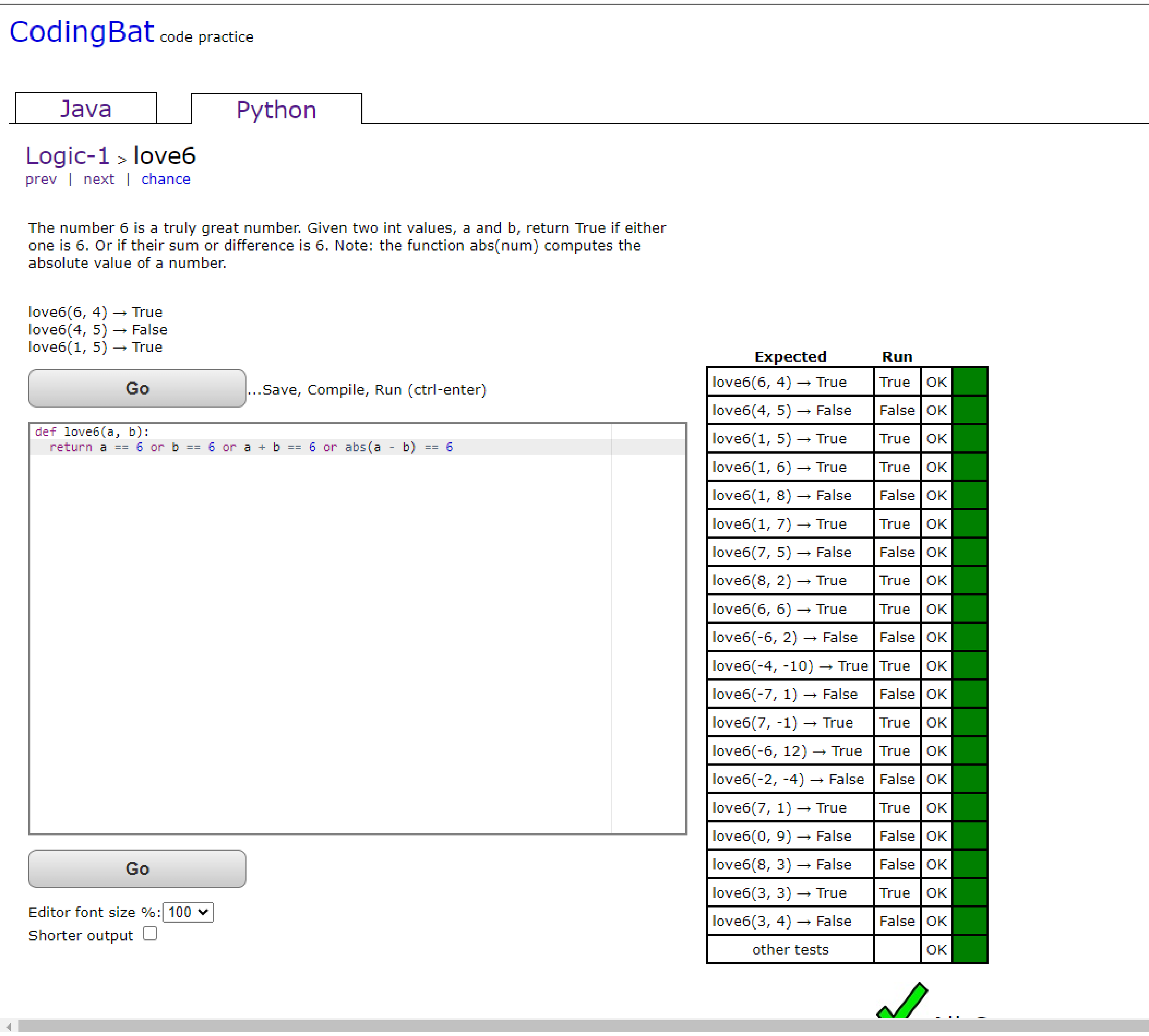
✓ All Correct

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem



CodingBat code practice

Logic-1 > love6

The number 6 is a truly great number. Given two int values, a and b, return True if either one is 6. Or if their sum or difference is 6. Note: the function abs(num) computes the absolute value of a number.

love6(6, 4) → True
love6(4, 5) → False
love6(1, 5) → True

```
def love6(a, b):
  return a == 6 or b == 6 or a + b == 6 or abs(a - b) == 6
```

Editor font size %: 100
Shorter output ☐

Expected	Run		
love6(6, 4) → True	True	OK	🟩
love6(4, 5) → False	False	OK	🟩
love6(1, 5) → True	True	OK	🟩
love6(1, 6) → True	True	OK	🟩
love6(1, 8) → False	False	OK	🟩
love6(1, 7) → True	True	OK	🟩
love6(7, 5) → False	False	OK	🟩
love6(8, 2) → True	True	OK	🟩
love6(6, 6) → True	True	OK	🟩
love6(-6, 2) → False	False	OK	🟩
love6(-4, -10) → True	True	OK	🟩
love6(-7, 1) → False	False	OK	🟩
love6(7, -1) → True	True	OK	🟩
love6(-6, 12) → True	True	OK	🟩
love6(-2, -4) → False	False	OK	🟩
love6(7, 1) → True	True	OK	🟩
love6(0, 9) → False	False	OK	🟩
love6(8, 3) → False	False	OK	🟩
love6(3, 3) → True	True	OK	🟩
love6(3, 4) → False	False	OK	🟩
other tests		OK	🟩

CodingBat code practice

Logic-1 > in1to10

prev | next | chance

Given a number n, return True if n is in the range 1..10, inclusive. Unless outside_mode is True, in which case return True if the number is less or equal to 1, or greater or equal to 10.

in1to10(5, False) → True
in1to10(11, False) → False
in1to10(11, True) → True

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

```
def in1to10(n, outside_mode):
  if outside_mode:
    return n <= 1 or n >= 10
  else:
    return 1 <= n <= 10
```

Go

Editor font size %: 100
Shorter output ☐

Expected	Run		
in1to10(5, False) → True	True	OK	🟩
in1to10(11, False) → False	False	OK	🟩
in1to10(11, True) → True	True	OK	🟩
in1to10(10, False) → True	True	OK	🟩
in1to10(10, True) → True	True	OK	🟩
in1to10(9, False) → True	True	OK	🟩
in1to10(9, True) → False	False	OK	🟩
in1to10(1, False) → True	True	OK	🟩
in1to10(1, True) → True	True	OK	🟩
in1to10(0, False) → False	False	OK	🟩
in1to10(0, True) → True	True	OK	🟩
in1to10(-1, False) → False	False	OK	🟩
in1to10(-1, True) → True	True	OK	🟩
in1to10(99, False) → False	False	OK	🟩
in1to10(-99, True) → True	True	OK	🟩
other tests		OK	🟩

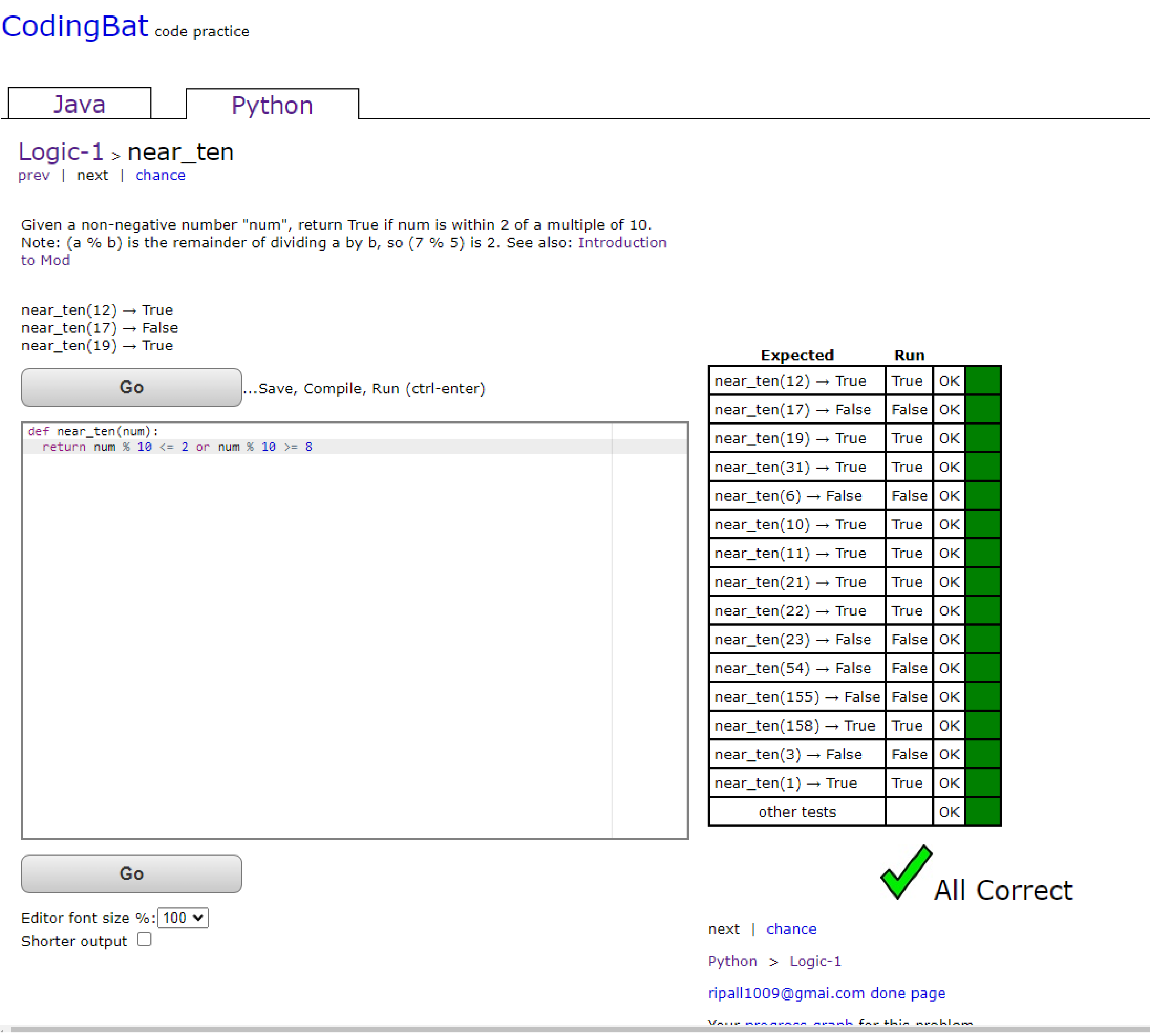
✓ All Correct

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem



CodingBat code practice

Java | Python

Logic-1 > near_ten

prev | next | chance

Given a non-negative number "num", return True if num is within 2 of a multiple of 10. Note: (a % b) is the remainder of dividing a by b, so (7 % 5) is 2. See also: Introduction to Mod

near_ten(12) → True
near_ten(17) → False
near_ten(19) → True

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

```
def near_ten(num):
  return num % 10 <= 2 or num % 10 >= 8
```

Go

Editor font size %: 100
Shorter output

Expected	Run		
near_ten(12) → True	True	OK	🟩
near_ten(17) → False	False	OK	🟩
near_ten(19) → True	True	OK	🟩
near_ten(31) → True	True	OK	🟩
near_ten(6) → False	False	OK	🟩
near_ten(10) → True	True	OK	🟩
near_ten(11) → True	True	OK	🟩
near_ten(21) → True	True	OK	🟩
near_ten(22) → True	True	OK	🟩
near_ten(23) → False	False	OK	🟩
near_ten(54) → False	False	OK	🟩
near_ten(155) → False	False	OK	🟩
near_ten(158) → True	True	OK	🟩
near_ten(3) → False	False	OK	🟩
near_ten(1) → True	True	OK	🟩
other tests		OK	🟩

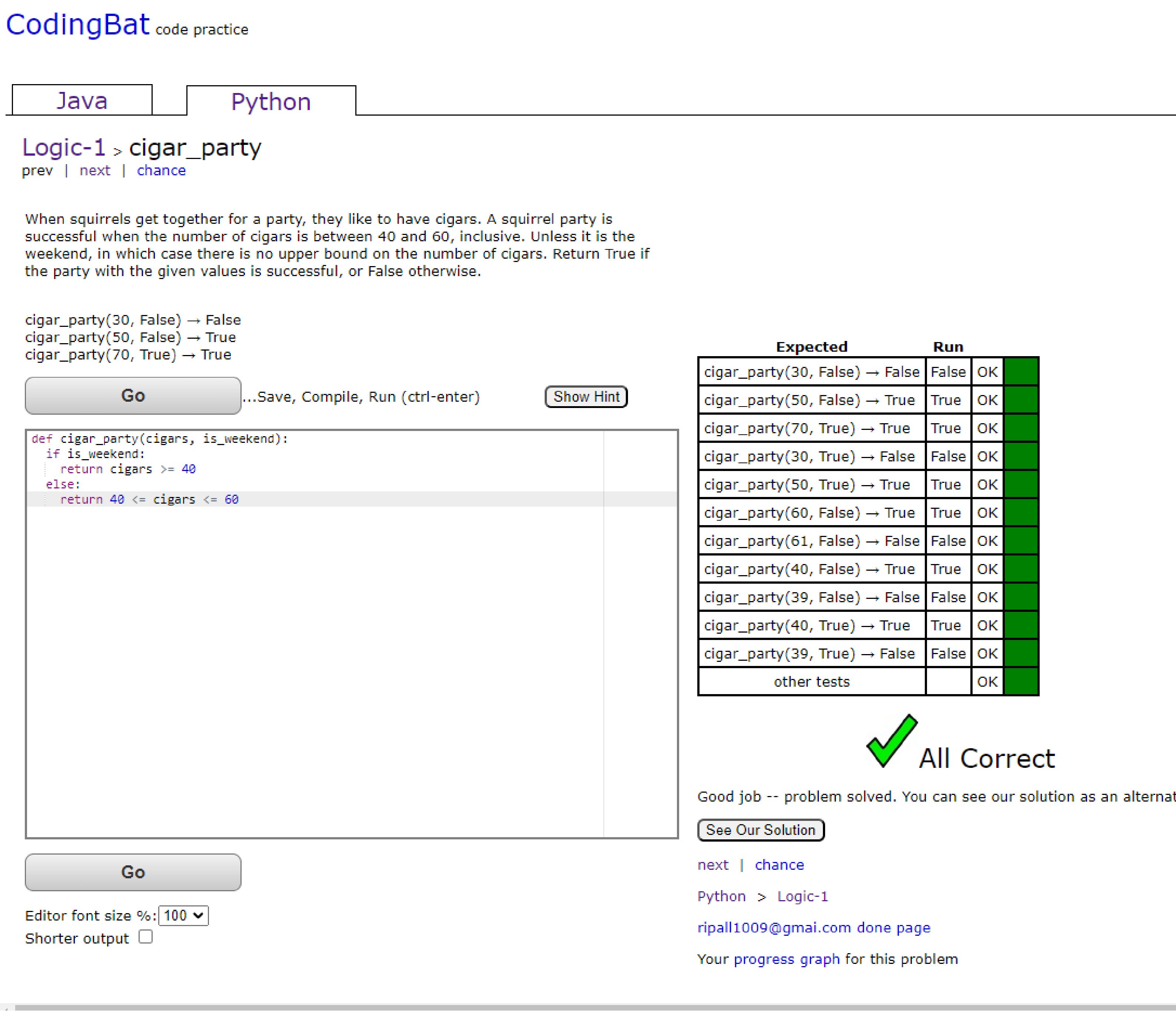
✓ All Correct

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem


CodingBat code practice

Java | Python

Logic-1 > cigar_party
prev | next | chance

When squirrels get together for a party, they like to have cigars. A squirrel party is successful when the number of cigars is between 40 and 60, inclusive. Unless it is the weekend, in which case there is no upper bound on the number of cigars. Return True if the party with the given values is successful, or False otherwise.

cigar_party(30, False) → False
cigar_party(50, False) → True
cigar_party(70, True) → True

```
def cigar_party(cigars, is_weekend):
  if is_weekend:
    return cigars >= 40
  else:
    return 40 <= cigars <= 60
```

Editor font size %: 100
Shorter output ☐

Expected	Run		
cigar_party(30, False) → False	False	OK	🟩
cigar_party(50, False) → True	True	OK	🟩
cigar_party(70, True) → True	True	OK	🟩
cigar_party(30, True) → False	False	OK	🟩
cigar_party(50, True) → True	True	OK	🟩
cigar_party(60, False) → True	True	OK	🟩
cigar_party(61, False) → False	False	OK	🟩
cigar_party(40, False) → True	True	OK	🟩
cigar_party(39, False) → False	False	OK	🟩
cigar_party(40, True) → True	True	OK	🟩
cigar_party(39, True) → False	False	OK	🟩
other tests		OK	🟩

✓ All Correct

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

next | chance

Python > Logic-1

ripall1009@gmai.com done page

Your progress graph for this problem