Tecnológico de Monterrey

Análisis, diseño y construcción de software

Lab 2.1 - Python Boost

Profesor

Dr. Gerardo Padilla Zarate

Integrantes

Matricula	Nombre
A01220356	Paul Iván Gallegos Bernal
A00354877	Héctor Gabriel Olagues Torres

4–Python Failing

Description

You need to create two functions to substitute str() and int(). A function called int_to_str() that converts integers into strings, and a function called str_to_int() that converts strings into integers.

Test Cases and Evidence

- Provide 10 test cases for each function

Code

```python
num_string_arr = ['0','1','2','3','4','5','6','7','8','9']

def int_to_str(number):
  if number < 0:
    return 'Not allowing negative numbers'

  if number == 0:
    return '0'

  num_str = ''
  while number > 0:
    digit = number % 10
    number = int(number / 10)
    num_str =  num_string_arr[digit] + num_str

  return num_str

def str_to_int(string_number):
  number = 0
  for character in string_number:
      number = number * 10
      digit = num_string_arr.index(character)
      number = number + digit

  return number
```

Test cases and evidence:

```
int_test_values = [-1, 0, 1, 5, 12345, 98765, 4116, 6622, 999, 00015]

print('int_to_str Test Cases ({0}):'.format(len(int_test_values)))
print('')

index = 0
for value in int_test_values:
  index = index + 1
  result = int_to_str(value)
  print('Test {0}: Value {1} converted in string: {2}'.format(index, value, result))

print('')

str_test_values = ['0', '1', '26', '536', '000452', '2626', '17771', '12677772234', '0101010', '999999']
print('str_to_int Test Cases ({0}):'.format(len(int_test_values)))
print('')


index = 0
for value in str_test_values:
  index = index + 1
  result = str_to_int(value)
  print('Test {0}: Value {1} converted to int: {2}'.format(index, value, result))
```

```
int_to_str Test Cases (10):

Test 1: Value -1 converted in string: Not allowing negative numbers
Test 2: Value 0 converted in string: 0
Test 3: Value 1 converted in string: 1
Test 4: Value 5 converted in string: 5
Test 5: Value 12345 converted in string: 12345
Test 6: Value 98765 converted in string: 98765
Test 7: Value 4116 converted in string: 4116
Test 8: Value 6622 converted in string: 6622
Test 9: Value 999 converted in string: 999
Test 10: Value 13 converted in string: 13

str_to_int Test Cases (10):

Test 1: Value 0 converted to int: 0
Test 2: Value 1 converted to int: 1
Test 3: Value 26 converted to int: 26
Test 4: Value 536 converted to int: 536
Test 5: Value 000452 converted to int: 452
Test 6: Value 2626 converted to int: 2626
Test 7: Value 17771 converted to int: 17771
Test 8: Value 12677772234 converted to int: 12677772234
Test 9: Value 0101010 converted to int: 101010
Test 10: Value 999999 converted to int: 999999
[Finished in 0.1s]
```

5-Profit

Description

You work for a manufacturer, and have been asked to calculate the total profit made on the sales of a product. You are given a dictionary containing the cost price per unit(in dollars), sell price per unit(in dollars), and the starting inventory. Return the total profit made, rounded to the nearest dollar.

Test Cases and Evidence

- Provide10 test cases

Code

```
def profit(data):
    total_profit = (data['sell_price'] - data['cost_price']) * data['inventory']
    return int(round(total_profit))
```

Test cases and evidence:

```
test_cases_cost_lower = [
    {"cost_price": 0.00,"sell_price": 2.00,"inventory": 100},
    {"cost_price": 1.00,"sell_price": 2.00,"inventory": 100},
    {"cost_price": 32.67,"sell_price": 45.00,"inventory": 1200}
]

test_cases_cost_higher = [
    {"cost_price": 1,"sell_price": 0.00,"inventory": 1200},
    {"cost_price": 2,"sell_price": 0.00,"inventory": 1200},
    {"cost_price": 2,"sell_price": 0.00,"inventory": 1200}
]

test_cases_same = [
    {"cost_price": 65.2,"sell_price": 65.2,"inventory": 1000},
    {"cost_price": 65.2,"sell_price": 65.2,"inventory": 453}
]


test_cases_no_inventory = [
    {"cost_price": 100,"sell_price": 1.00,"inventory": 0},
    {"cost_price": 1.5,"sell_price": 100.00,"inventory": 0}
]
```

```
index = 1

def printTestCases(test_cases):
    global index
    for test in test_cases:
        print('Testcase: {0}:'.format(index))
        print('Input: {0}:'.format(test))
        print('Output: {0}'.format(profit(test)))
        print('')
        index = index + 1
```

```
print('#####################################')
print('Cost price is lower than sell price:')
printTestCases(test_cases_cost_lower)

print('#####################################')
print('Cost price is higher than sell price:')
printTestCases(test_cases_cost_higher)

print('#####################################')
print('Cost price is the same as sell price:')
printTestCases(test_cases_same)

print('#####################################')
print('No inventory:')
printTestCases(test_cases_no_inventory)
```

```
####################################
Cost price is lower than sell price:
Testcase: 1:
Input: {'sell_price': 2.0, 'inventory': 100, 'cost_price': 0.0}:
Output: 200

Testcase: 2:
Input: {'sell_price': 2.0, 'inventory': 100, 'cost_price': 1.0}:
Output: 100

Testcase: 3:
Input: {'sell_price': 45.0, 'inventory': 1200, 'cost_price': 32.67}:
Output: 14796

####################################
Cost price is higher than sell price:
Testcase: 4:
Input: {'sell_price': 0.0, 'inventory': 1200, 'cost_price': 1}:
Output: -1200

Testcase: 5:
Input: {'sell_price': 0.0, 'inventory': 1200, 'cost_price': 2}:
Output: -2400

Testcase: 6:
Input: {'sell_price': 0.0, 'inventory': 1200, 'cost_price': 2}:
Output: -2400
```

```
####################################
Cost price is the same as sell price:
Testcase: 7:
Input: {'sell_price': 65.2, 'inventory': 1000, 'cost_price': 65.2}:
Output: 0

Testcase: 8:
Input: {'sell_price': 65.2, 'inventory': 453, 'cost_price': 65.2}:
Output: 0

####################################
No inventory:
Testcase: 9:
Input: {'sell_price': 1.0, 'inventory': 0, 'cost_price': 100}:
Output: 0

Testcase: 10:
Input: {'sell_price': 100.0, 'inventory': 0, 'cost_price': 1.5}:
Output: 0

[Finished in 0.1s]
```