Activity

practising returning data using functions

model answers

**Challenge 1**

**Write a short 2 question quiz that uses a function to evaluate each answer and return a score.**

# Getting Data from Functions

# Challenge 1 (Two-item Quiz)

#

# date: November 2016

score = 0

print('\nTwo-item Quiz Example')

def get\_score(question, answer, score):

print('\n'+ question)

user\_response = input('Answer: ')

if user\_response == answer:

score += 1

print('Well done')

else:

print('Sorry, you got it wrong')

return score

score = get\_score('What do you call the scope of a variable that can be accessed anywhere in the code?', 'global', 0)

score = get\_score('What Python statement allows for throwing back of a value used in the function to the calling statement?', 'return', score)

print('\nYour score: {0}'.format(score))

'''

# assertion

inputs: global, hello

Two-item Quiz Example

What do you call the scope of a variable that can be accessed anywhere in the code?

Answer: global

Well done

What Python statement allows for throwing back of a value used in the function to the calling statement?

Answer: hello

Sorry, you got it wrong

Your score: 1

'''

**Challenge 2**

**Create a simple calculator that just sums 2 numbers entered by the user. Use the function to return the calculated answer to the calling code.**

# Getting Data from Functions

# Challenge 2 (Use a function to get the sum of two numbers)

#

# date: November 2016

score = 0

print('\nSumming Two Numbers')

def get\_sum(num1, num2):

sum = num1 + num2

return sum

num1 = int(input('\nEnter the first number: '))

num2 = int(input('\nEnter the second number: '))

print('\nThe sum of {0} and {1} is {2}'.format(num1, num2, get\_sum(num1, num2)))

'''

# assertion

input 4 and 3

output:

Summing Two Numbers

Enter the first number: 4

Enter the second number: 3

The sum of 4 and 3 is 7

'''

**Challenge 3**

**This task is repeated from earlier in the course. Use your current skills to see if you can write more concise code using a function with a return statement.**

**A parking meter charges $4 for the first 2 hours, then $3 for every hour thereafter. Write a program that displays a welcome message and instructions, then prompts the user for the number of hours that they need to park. The program should display the calculated cost of parking.**

# Getting Data from Functions

# Challenge 3 (Parking Cost Calculator)

#

# date: November 2016

score = 0

print('\nParking Cost Calculator')

print('\nThis program calculates the cost of parking based on the following formula:'

'\n $4 for the first 2 hours, then $3 for every hour thereafter.')

def get\_parking\_cost(hours):

cost = (hours - 2) \* 3 + 4

return cost

hours = int(input('\nEnter the number of parking hours (integer only): '))

print('\nThe cost of parking for {0} hours is {1}'.format(hours, get\_parking\_cost(hours)))

'''

# assertion

input 4

output: 10

Parking Cost Calculator

This program calculates the cost of parking based on the following formula:

$4 for the first 2 hours, then $3 for every hour thereafter.

Enter the number of parking hours (integer only): 4

The cost of parking for 4 hours is 10

# assertion 2

input 1

output: 4

Parking Cost Calculator

This program calculates the cost of parking based on the following formula:

$4 for the first 2 hours, then $3 for every hour thereafter.

Enter the number of parking hours (integer only): 4

The cost of parking for 4 hours is 10

'''

**Challenge 4**

**list\_a = ["brown", "grey", "amber"]**

**list\_b = ["red", "green"]**

**list\_c = ["purple"]**

**Create a program that has a stored list list\_d which contains the 3 lists shown above. The program must check lists a to c. For any list that contains less than 3 elements, the program must prompt for user input and use the data entered to fill the list.**

# Getting Data from Functions

# Challenge 4 (Colour Lists)

#

# date: November 2016

print('\nColour Lists\n')

list\_a = ["brown", "grey", "amber"]

list\_b = ["red", "green"]

list\_c = ["purple"]

col\_list = [list\_a, list\_b, list\_c]

def get\_colour(colour\_list):

ctr = len(colour\_list)

while ctr < 3:

colour = input('Please enter another colour to complete the list {0}: '.format(colour\_list))

ctr += 1

colour\_list.append(colour)

return colour\_list

for clist in col\_list:

get\_colour(clist)

print('\nThe 3-colour lists are:')

for clist in col\_list:

print(clist)

'''

# assertion

input: yellow

input: green, blue

output:

Colour Lists

Please enter another colour to complete the list ['red', 'green']: yellow

Please enter another colour to complete the list ['purple']: green

Please enter another colour to complete the list ['purple', 'green']: blue

The 3-colour lists are:

['brown', 'grey', 'amber']

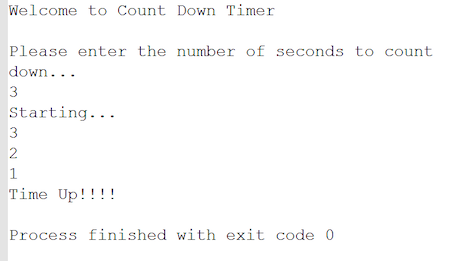
['red', 'green', 'yellow']

['purple', 'green', 'blue']

'''

**Challenge 5**

**Write a program that produces an output similar that shown below.**

****

# Getting Data from Functions

# Challenge 5 (Count Down Timer)

#

# date: November 2016

from time import sleep

print('\nWelcome to Count Down Timer\n')

def get\_timer():

seconds = int(input('Please enter the number of seconds to count down...\n'))

return seconds

def get\_countdown\_exit\_code(seconds):

start\_time = seconds

print('Starting...')

for ctr in range(seconds):

current\_time = start\_time - ctr

print(current\_time)

sleep(1)

print('Time Up!!!!')

return 0

print('\nProcess finished with exit code {0}'.format(get\_countdown\_exit\_code(get\_timer())))

'''

#assertions

Input: 5

Output:

Welcome to Count Down Timer

Please enter the number of seconds to count down...

5

Starting...

5

4

3

2

1

Time Up!!!!

Process finished with exit code 0

'''

**Challenge 6**

**Write a Python function to find the Max of 3 stored numbers.**

# Getting Data from Functions

# Challenge 6 (Max of 3 Numbers)

#

# date: November 2016

print('\nFinding max of three numbers\n')

numlist = []

def get\_number\_list(numlist):

ctr = len(numlist)

while ctr < 3:

num = float(input('Please enter a number: '))

if num.is\_integer():

num = int(num)

numlist.append(num)

ctr += 1

return numlist

def get\_max(numlist):

max = numlist[0]

ctr = 1

while ctr < len(numlist):

if numlist[ctr] > max:

max = numlist[ctr]

ctr += 1

return max

numlist = get\_number\_list([])

print('\n{0} is the largest of the three numbers: {1}'.format(get\_max(numlist), numlist))

'''

#assumptions

Input, 56, 34, 78

Output:

Finding max of three numbers

Please enter a number: 56

Please enter a number: 34

Please enter a number: 78

78 is the largest of the three numbers: [56, 34, 78]

'''

**Challenge 7**

**Write a Python function that takes a non-negative integer as its parameter. The function should calculate the factorial of that number and return the result.**

# Getting Data from Functions

# Challenge 7 (Factorial)

#

# date: November 2016

print('\nFinding the factorial of a number')

def get\_factorial(num):

ctr, fact = 1, 1

while ctr <= num:

fact \*= ctr

ctr += 1

return fact

num = 0

while not int(num) > 0:

num = input('\nPlease enter a positive integer: ')

print('\nThe factorial of {0} is {1}'.format(num, get\_factorial(int(num))))

'''

#assertions

Input: 5

Output:

Finding the factorial of a number

Please enter a positive integer: 5

The factorial of 5 is 120

'''

**Challenge 8**

**Write a function that takes 3 integer parameters and returns boolean. If the first number is within the range between the second and third numbers (inclusive) then the function should return True, otherwise it should return false.**

# Getting Data from Functions

# Challenge 8 (Number within Range)

#

# date: November 2016

print('\nFinding if first number is between the second and third numbers\n')

numlist = []

def get\_number\_list(numlist):

ctr = len(numlist)

while ctr < 3:

num = float(input('Please enter a number: '))

if num.is\_integer():

num = int(num)

numlist.append(num)

ctr += 1

return numlist

def get\_within\_range\_truth\_value(numlist):

if numlist[2] > numlist[1]:

if (numlist[0] >= numlist[1] and

numlist[0] <= numlist[2]):

return True

else:

return False

else:

if (numlist[0] >= numlist[2] and

numlist[0] <= numlist[1]):

return True

else:

return False

numlist = get\_number\_list([])

print('\nIt is {0} that {1} is within the range of {2} and {3}.'.format(str(get\_within\_range\_truth\_value(numlist)).lower(), numlist[0], numlist[1],numlist[2]))

'''

#assertions

Input: 45, 10, 67

Output:

Finding if first number is between the second and third numbers

Please enter a number: 45

Please enter a number: 10

Please enter a number: 67

It is true that 45 is within the range of 10 and 67.

#assertion 2

Input: 10, 34, 100

Output:

Finding if first number is between the second and third numbers

Please enter a number: 10

Please enter a number: 34

Please enter a number: 100

It is false that 10 is within the range of 34 and 100.

'''

**Challenge 9**

**A stored list has 3 numbers in it. Write a program that gets an integer from the user and then returns the number from the list that is closest to the users number. If list elements are the same distance from the number entered then return any one element.**

**The next 2 challenges require functions that return multiple values. Since this is not possible, consider using any suitable container type to store the values, then return the container instead.**

# Getting Data from Functions

# Challenge 9 (Closest Number)

#

# date: November 2016

print('\nFinding the Closest Number')

numlist = [8, 28, 50]

num = float(input('\nPlease enter a number: '))

if num.is\_integer():

num = int(num)

def get\_closest\_number(num, numlist):

ctr = 1;

closest = numlist[0]

diff = abs(numlist[0] - num)

while ctr < len(numlist):

nDiff = abs(numlist[ctr] - num)

if (nDiff < diff):

diff = nDiff

closest = numlist[ctr]

ctr += 1

return closest

print('\nList of numbers to compare with: {0}'.format(numlist))

print('\n{0} is closest to {1}.'.format(num, get\_closest\_number(num, numlist)))

'''

#assertions

Input: 56

Output:

Finding the Closest Number

Please enter a number: 56

List of numbers to compare with: [8, 28, 50]

56 is closest to 50.

#Assertion 2

Input: 27

Output:

Finding the Closest Number

Please enter a number: 27

List of numbers to compare with: [8, 28, 50]

27 is closest to 28.

'''

**Challenge 10**

**Write a Python function that accepts a string and calculates the number of upper case letters and lower case letters.**

# Getting Data from Functions

# Challenge 10 (Count upper and lower case characters)

#

# date: November 2016

print('\nCounting Characters\n')

word = input("Please enter a string: ")

def get\_character\_count(word):

uCtr, lCtr, oCtr = 0, 0, 0

for key, value in enumerate(word):

if value.isupper():

uCtr += 1

elif value.islower():

lCtr += 1

else:

oCtr += 1

return {'Upper case letters':uCtr, 'Lower case letters':lCtr, 'Other characters':oCtr}

print('\nCharacter Counts:\n')

counter\_dict = get\_character\_count(word)

for key,value in counter\_dict.items():

print('{0}: {1}'.format(key, value))

'''

#assertions

Input: IlikeTIGERS

Output:

Counting Characters

Please enter a string: IlikeTIGERS

Character Counts:

Lower case letters: 4

Upper case letters: 7

Other characters: 0

#assertion 2

Input: I love cats

Output:

Please enter a string: I love cats

Character Counts:

Other characters: 2

Upper case letters: 1

Lower case letters: 8

'''

**Challenge 11**

**Write a function that takes a stored list of integers and returns the values that are less than or equal to the list average.**

# Getting Data from Functions

# Challenge 11 (Values less than the average)

#

# date: November 2016

print('\nDisplaying values less than the average in a list\n')

numlist = [23, 45, 62, 33, 76, 13, 24, 66]

def get\_average(numlist):

summation = 0

for num in numlist:

summation += num

return summation / len(numlist)

def get\_lower\_nums(average, numlist):

lower\_nums = []

for num in numlist:

if num < average:

lower\_nums.append(num)

return lower\_nums

average = get\_average(numlist)

lower\_nums = get\_lower\_nums(average, numlist)

print('Average: {0}\nValues less than the average: {1} '.format(average, lower\_nums))

'''

#assertions

Displaying values less than the average in a list

Average: 42.75

Values less than the average: [23, 33, 13, 24]

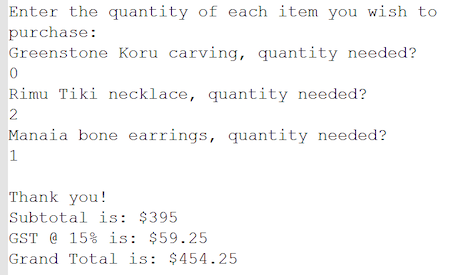
'''

**Challenge 12**

**Here is some text from a shopping advert.**

**Today's Maori souvenir specials (excl. GST) are:  
  
Greenstone Koru carving @ just $265,  
Rimu Tiki necklace is a bargain at only $42  
Manaia bone earrings for only $311**

**You must write some code for the shop's online store. Write a program that produces output similar to the one shown below.**

****

# Getting Data from Functions

# Challenge 12 (Online Shopping)

#

# date: November 2016

print('\nOnline Store Application\n')

store = {'Greenstone Koru carving': 265, 'Rimu Tiki necklace': 42, 'Manaia bone earrings': 311}

print('Enter the quantity of each item you wish to purchase: \n')

item\_quantity = {}

def get\_purchase\_total(store):

subtotal = 0

for key,value in store.items():

quantity = input(('{0} @ ${1}, quantity needed?\n'.format(key, value)))

subtotal += int(quantity) \* value

return subtotal

subtotal = get\_purchase\_total(store)

print('\nSubtotal is: ${0}'

'\nGST @15% is: ${1}'

'\nGrand Total is: ${2:0.2f}'.format(subtotal, .15\*subtotal, 1.15\*subtotal))

'''

#assertions

Input: 0,1,3

Output:

Online Store Application

Enter the quantity of each item you wish to purchase:

Rimu Tiki necklace @ $42, quantity needed?

0

Greenstone Koru carving @ $265, quantity needed?

1

Manaia bone earrings @ $311, quantity needed?

3

Subtotal is: $1198

GST @15% is: $179.7

Grand Total is: $1377.70

#assertion 2

Input: 1,0,2

Output:

Online Store Application

Enter the quantity of each item you wish to purchase:

Manaia bone earrings @ $311, quantity needed?

1

Greenstone Koru carving @ $265, quantity needed?

0

Rimu Tiki necklace @ $42, quantity needed?

2

Subtotal is: $395

GST @15% is: $59.25

Grand Total is: $454.25

'''

**Challenge 13**

**Write a Calculator program that meets the following requirements:**

* **The program displays an appropriate welcome message and user instructions.**
* **It prompts the user for their required operation, first number and second number.**
* **It performs the required operation using the 2 numbers and displays the result on screen.**
* **The program loops until the user enters "exit".**
* **Assume that the user enters only valid integers or "exit".**
* **The operations to be performed are add, subtract, multiply and divide.**

# Functions With Parameters

# Challenge 7: Calculator

#

# date: November 2016

print('\nWelcome to my Calculator Program!')

print('\nWhen asked for the operation, please enter add, subtract, multiply, divide, or exit.')

print('\nWhen you enter exit, the program will terminate.')

print('\nWhen you enter any of the valid operations, you be prompted for 2 numbers.')

# create an operations dictionary

operations = {'add':'+', 'subtract':'-', 'multiply':'\*', 'divide':'/', 'exit':'X'}

def input\_number(nth):

while True:

n1 = input('\nPlease enter the ' + nth + ' number: ')

if n1.is\_numeric():

return int(n1)

elif n1.lower() == 'exit':

return n1

else:

print('You need to enter an integer, try again')

def compute(op, n1 ,n2):

if op == 'add':

result = n1 + n2

elif op == 'subtract':

result = n1 - n2

elif op == 'multiply':

result = n1 \* n2

elif op == 'divide':

result = n1 / n2

else:

return "Error: Compute operation was not found."

return result

while True:

op = input('\nPlease enter the operation: ').lower()

if op in operations.keys():

if op != 'exit':

n1 = input\_number('first')

if n1 == 'exit':

break

n2 = input\_number('second')

if n2 == 'exit':

break

result = compute(op, n1, n2)

print('\n', n1, operations[op], n2, '=', result)

else:

print("Thanks for using the calculator")

break

else:

print('You need to enter one of these: add, subtract, multiply, divide, or exit')

'''

# test case assertion 1

print("\n\nTest case assertion 1 \n\n")

print("3 + 5 = 8")

'''