Assignment 4 - Functions

23 pts

In [20]:
```
# create a function to return your name and studentID as a tuple
# then call the function to print out the values
def register(name, studentID):
    return name, studentID

student_name, student_id = register("Yong Seung Rho", "W0447442")
print(f"'Name: {student_name}, StudentID: {student_id}'")
```

```
'Name: Yong Seung Rho, StudentID: W0447442'
```

'Name: Jane Doe, StudentID: w123456'

Part A - Small Bits (11pts)

For each item below, determine the appropriate Python code to generate the desired output.

In [11]:
```
# create a function, called 'calc_restaurant_bill', that accepts a cost, a
# tax rate and a tip percentage. the function will return the total cost and
# the tip amount as a tuple. then, call the function to determine the bill and
# tip for a patron that orders the $9.99 daily special with 15% tax and an 18%
# tip.

# 2 pts
def calc_restaurant_bill(cost, tax_rate, tip):
    total_cost = cost + (cost * tax_rate / 100)
    tip = total_cost * (tip / 100)
    return total_cost, tip

# determine the bill and tip for a patron that orders the $9.99 daily special
# with 15% tax and an 18% tip.
patron_cost, patron_tip = calc_restaurant_bill(9.99, 15, 18)
print(f"Cost: ${patron_cost:.2f}, Tip: ${patron_tip:.2f}")
```

```
Cost: $11.49, Tip: $2.07
```

In [18]:
```
# create a function, called 'random_letter', that returns a random letter of the
# alphabet using the given string. call the function to generate 5 random
# letters, on a single line, separated by spaces
# 2 pts

import random

ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# put your code here
letters = []
def random_letter():
    letter = random.choices(ALPHABET)
    return letter[0]

# generate 5 random letters, on a single line, separated by spaces
letters = list()
for i in range(0, 5):
    char = random_letter()
    letters.append(char)
print(" ".join(letters))
```

```
Z K L E O
```

In [13]:
```
# expand on the previous question by creating a new function, called
# 'generate_password', to generate random passwords using calls to the previous
# function. the function should take a "length" parameter and return a password
# of that given length, containing random letters. call the new function to
# generate passwords with three different lengths (e.g. 8, 10, 12)
# 2 pts
def generate_password(length):
    letters = list()
    for i in range(0, length):
        char = random_letter()
        letters.append(char)
    password = "".join(letters)
    return password

# generate passwords with three different lengths (e.g. 8, 10, 12)
print(generate_password(8), generate_password(10), generate_password(12))
```

```
TRBDYGIC FWRHVAVBAE ALNIOCAWWRHQ
```

In [14]:
```
# create a function, called 'convert_from_cad', that converts between Canadian
# dollars and other currencies
# - the program should support CAD->USD, CAD->EUR and CAD->JPY
# - you can obtain the current exchange rates from "xe.com"
# - the exchange rates should be stored as a global dictionary
# - the function should allow two parameters,
#   one for the CAD amount and one for the other currency name (e.g. 'USD')
# call the function three times to display the currency exchanges for $1 CAD.
# 3 pts
exchange_rates = {"USD":0.768639, "EUR":0.650259, "JPY":80.9347}
def convert_from_cad(cad, currency_name):
    for key, value in exchange_rates.items():
        if key == currency_name:
            return cad * value
    return 0

print(f"$1 CAD->USD: {convert_from_cad(1, 'USD')}")
print(f"$1 CAD->EUR: {convert_from_cad(1, 'EUR')}")
print(f"$1 CAD->JPY: {convert_from_cad(1, 'JPY')}")
```

```
$1 CAD->USD: 0.768639
$1 CAD->EUR: 0.650259
$1 CAD->JPY: 80.9347
```

Part B - Big Bytes! (12pts)

The following are more challenging questions. Be patient when tackling these!

In [19]:
```
# expand on the previous question by adding a function, called
# 'decimal_to_dollars',  that can convert from a decimal amount of Canadian
# dollars to an equivalent amount of loonies, quarters, dimes and nickels.
# Note: as pennies no longer exist, you will round up/down to the nearest nickel.
# call the function to display the change for $2.73 CAD.
# 5 pts
def get_quantity(amount, change):
    quantity = 0
    if amount >= change:
        quantity = int(amount / change)
        amount = amount - quantity * change
    return quantity, amount

def decimal_to_dollars(amount):
    # convert from a decimal amount of Canadian
    # dollars to an equivalent amount of loonies, quarters, dimes and nickels.
    loonies, r = get_quantity(amount, 1)
    t_quarters, r = get_quantity(r, 0.25)
    dimes, r = get_quantity(r, 0.1)
    nickels, r = get_quantity(r, 0.05)

    # round up/down to the nearest nickel
    if r > 0.02:
        nickels = nickels + 1
    return loonies, t_quarters, dimes, nickels

# call the function to display the change for $2.73 CAD
l, q, d, n = decimal_to_dollars(2.73)
print(f"The change for $2.73 CAD: loonies: {l}, t_quarters: {q}, dimes: {d}, nickels: {n}")
```

```
The change for $2.73 CAD: loonies: 2, t_quarters: 2, dimes: 2, nickels: 1
```

In [16]:
```
# create a function, called 'add_score', to manage the top 10 list of a
# game's high scores
# the scores will be stored in a global list called "top_10_scores"
# the function will update the list by passing a new score as a parameter.
# the function will then add the new score if it is higher than a previous one,
# and removing any scores, below the top 10, from the list
# test the new function by calling it 15 times with random high scores between
# 1 and 10. display all the scores generated and then print out the resulting
# top 10 list
# note: you can sort a list using the "sort()" function (e.g. my_list.sort())
# 4pts
import random

top_10_scores = []

def add_score(score):
    if len(top_10_scores) < 10:
        # adds a new score
        top_10_scores.append(score)
    else:
        #if it is higher than a previous one, and removing any scores, below the top 10, from the list
        if top_10_scores[0] < score:
            top_10_scores[0] = score

    # sort to find the lowest score in top_10_scores
    top_10_scores.sort()

# test the new function by calling it 15 times with random high scores between 1 and 10
print("scores: ", end ='')
for i in range(15):
    r = random.randint(1, 10)
    add_score(r)

    # display all the scores generated
    print(f"{r} ", end='')

# print out the resulting top 10 list
print()
print("top_10_scores:", top_10_scores)

# when test_question6 runs after executing above code, top_10_scores still has the previous scores
# initialize top_10_scores to be empty for next execution
```

```
top_10_scores = list()
```

```
scores: 3 7 8 6 9 3 1 1 6 10 6 5 3 1 7
top_10_scores: [3, 5, 6, 6, 6, 7, 7, 8, 9, 10]
```

In [17]:
```
# create a function, called 'is_prime', that can determine if a given number is
# a "prime number" or not (a prime number is an integer that is only evenly
# divisible by itself and 1. the number 1 is not itself a prime number though.)
# then, use the function to list all the prime numbers between 1 and 50 (inclusive).
# 3 pts
def is_prime(num):
    if num == 1:
        return False

    # determine if a given number is a "prime number" or not
    for i in range(2, num):
        if num % i == 0:
            return False
    return True

prime_numbers = []
for i in range(1, 50):
    # list all the prime numbers between 1 and 50
    if is_prime(i):
        prime_numbers.append(i)

print("The prime numbers between 1 and 50: ", prime_numbers)
```

```
The prime numbers between 1 and 50:  [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

In [ ]:
```

```