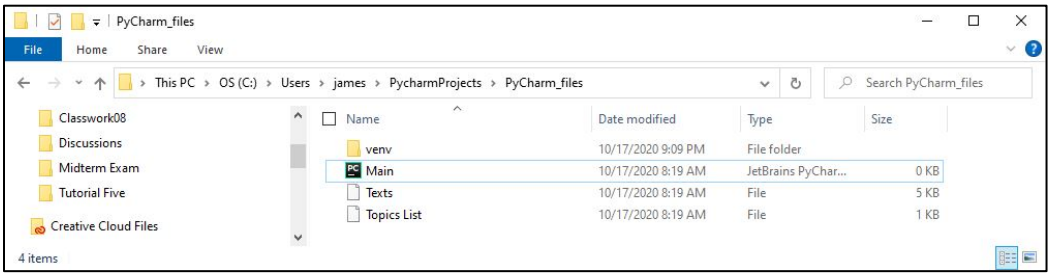
Exploring PyCharm—a Python IDE

Becoming Familiar with PyCharm:

1) Once PyCharm has been installed and while it is not running, use the PyCharm_Files.zip file to unzip its contents into the directory where PyCharm saves its project files. On Windows it is C:/Users/<yourname>/PyCharmProjects. The screenshot below reflects how the file structure should look after you unzip the contents:



2) a. Image 1 – Launch PyCharm.

a) Image 2 – Click on the PyCharm_files project listed on the top-left side of the window.

b) Image 3 – When PyCharm opens, click on the Project tab in the far-top-left side of window.

c) Image 4 – Right-click on 

in the project window for a prompt to create a new file, then choose New --> Python File to create a new .py file, enter Hello_World into the Name field of the pop-up window to name the new .py file, and press return to create Hello_World.py in the PyCharm_files project.

d) Image 5 – Type or copy/paste the next five lines of code into the Hello_World.py Python file.

```
# the hashtag sign # denotes a comment in Python code #
# print() function example 'Hello World!'… #
print('Hello World!')
print(' ')
print('Managing Data for Analytics')
```

The # denotes a comment line of code.

The print() function prints to the screen whatever variable is between the ().

e) Image 6 – Alt-Shift-F10 or choose Run--> Run 'Hello_World' from the command ribbon will execute the Python file Hello_World.py and the Run: window will launch across the bottom section of the PyCharm window. In the Run: window, you will see the strings 'Hello World!' and 'Managing Data for Analytics' printed on two separate lines with a blank ' ' line between the strings. Can you follow the logic of the Hello_World.py file? Congratulations on executing your first MIS 502 Python code.

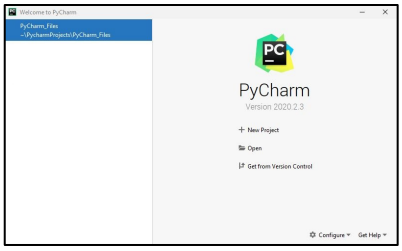


Image 1

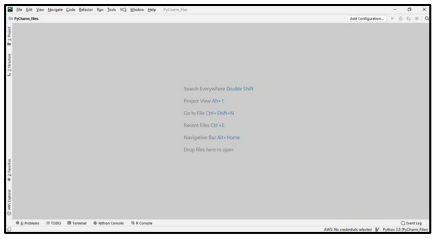


Image 2

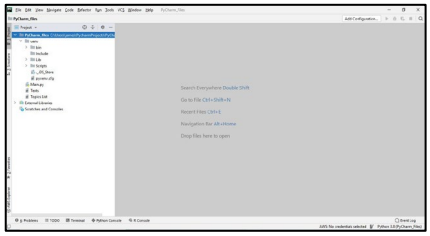


Image 3

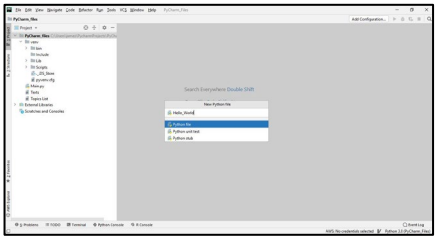


Image 4

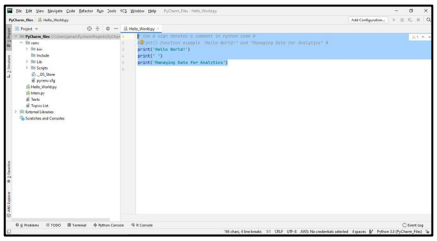


Image 5

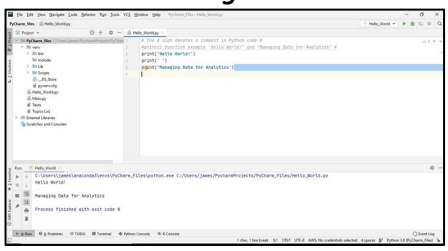


Image 6

The images above can be viewed at 400% magnification for details of the screen shots.

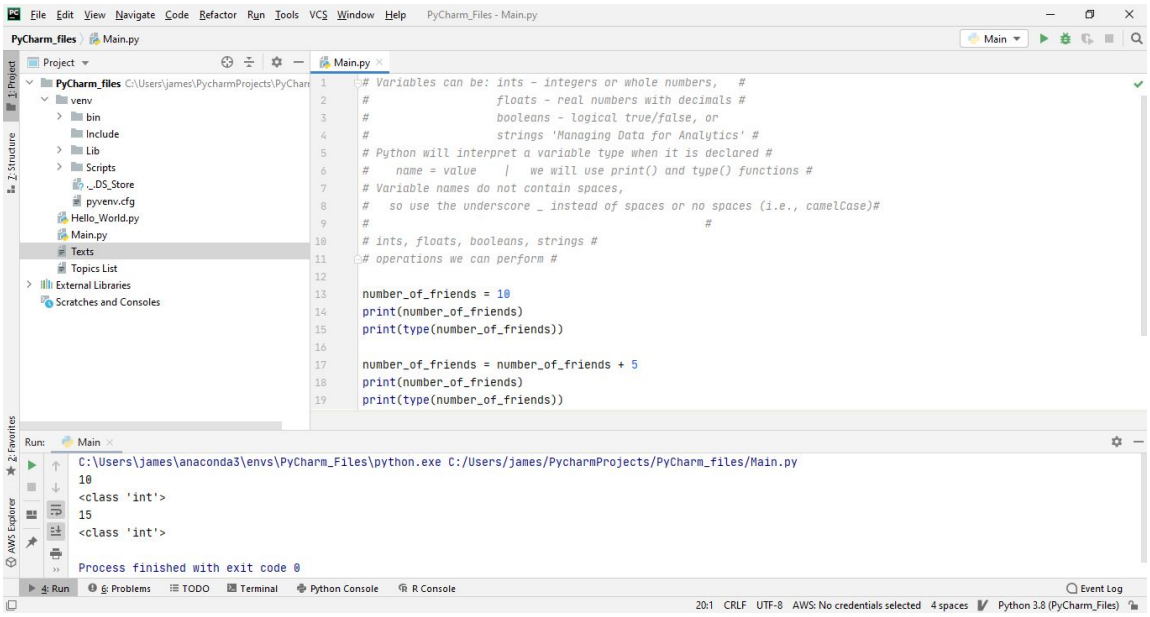
3) After you have run (i.e., executed) your version of the Hello_World.py Python file, take a screenshot of your PyCharm windows (e.g.,project, Hello_World.py, and Run:) as the deliverables of this section.

Declaring Variables in Python:

1) ints (Integers or whole numbers) – Copy the Python code below into Main.py and Run.

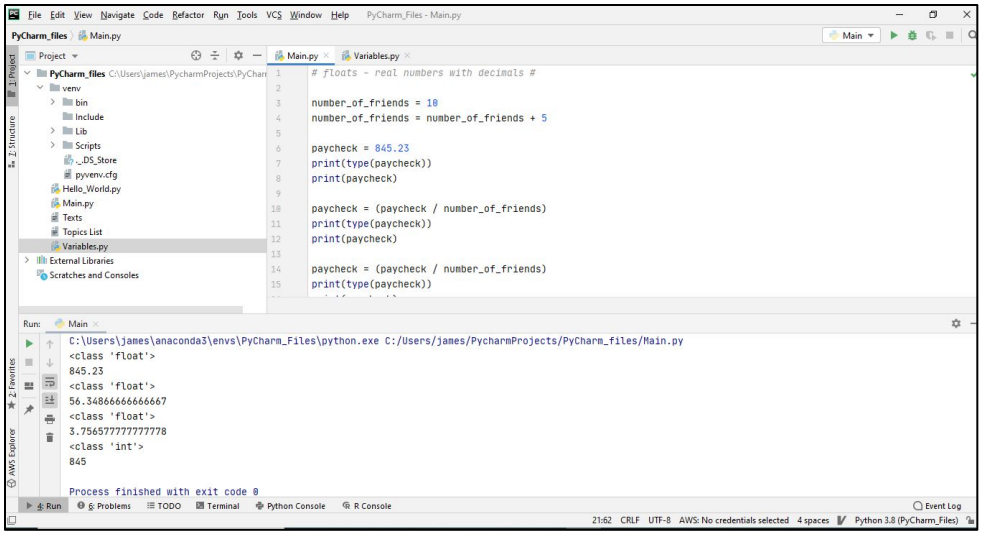
```
# Variables can be: ints - integers or whole numbers,    #
#                   floats - real numbers with decimals #
#                   booleans - logical true/false, or
#                   strings 'Managing Data for Analytics' #
# Python will interpret a variable type when it is declared #
#    name = value    |   we will use print() and type() functions #
# Variable names do not contain spaces, so use the             #
#   underscore _ instead of spaces or no spaces (i.e., camelCase)#
#                                                              #
# ints, floats, booleans, strings #
# ints - Integers #

number_of_friends = 10
print(number_of_friends)
print(type(number_of_friends))
```

```
number_of_friends = number_of_friends + 5
print(number_of_friends)
print(type(number_of_friends))
```



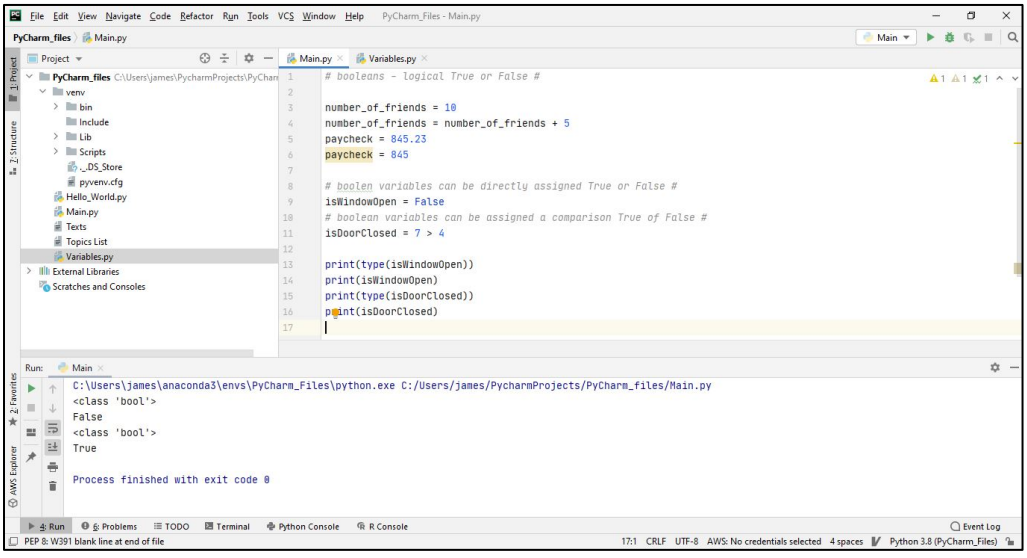
2) floats (real numbers with decimals) – Copy the Python code below into Main.py and Run.

```
# floats - real numbers with decimals #

number_of_friends = 10
number_of_friends = number_of_friends + 5

paycheck = 845.23
print(type(paycheck))
print(paycheck)

paycheck = (paycheck / number_of_friends)
print(type(paycheck))
print(paycheck)

paycheck = (paycheck / number_of_friends)
print(type(paycheck))
print(paycheck)

paycheck = 845
print(type(paycheck))
print(paycheck)
# what happened to the paycheck variable? #
# paycheck was reassigned a whole number and became an ints #
```



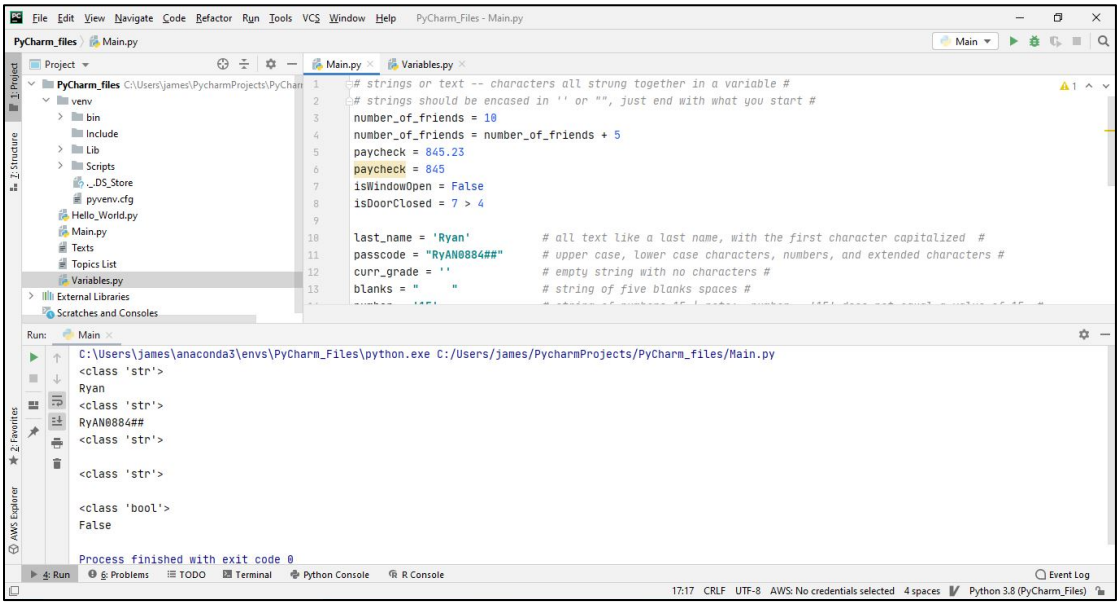
3) booleans (logical True or False) – Copy the Python code below into Main.py and Run.

```
# booleans - logical True or False #

number_of_friends = 10
number_of_friends = number_of_friends + 5
paycheck = 845.23
paycheck = 845

# boolen variables can be directly assigned True or False #
isWindowOpen = False
# boolean variables can be assigned a comparison True of False #
isDoorClosed = 7 > 4

print(type(isWindowOpen))
print(isWindowOpen)
print(type(isDoorClosed))
print(isDoorClosed)
```



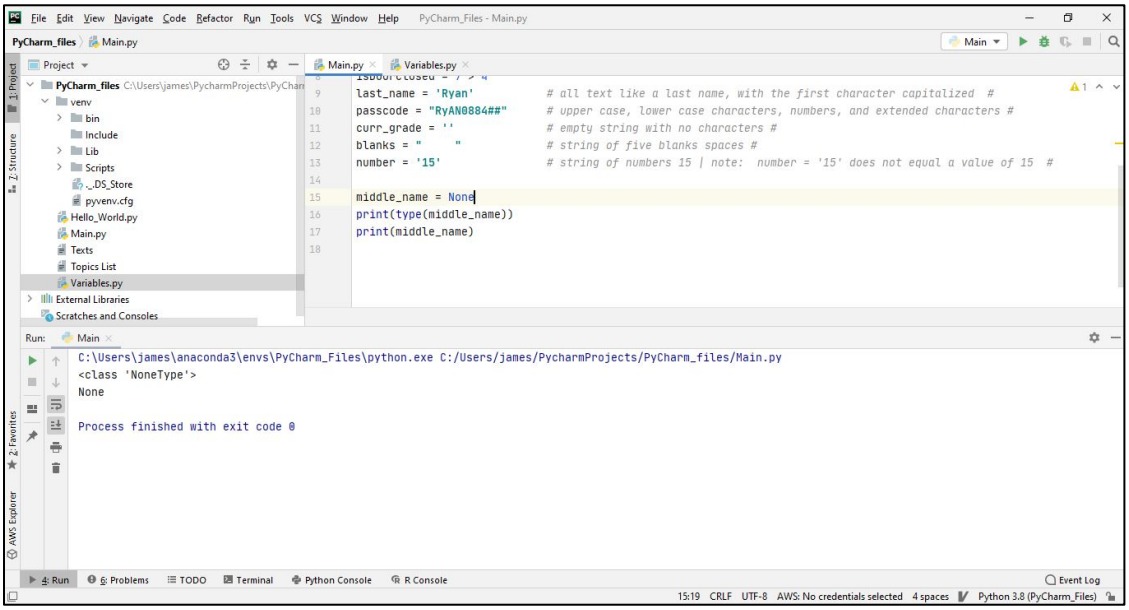
4) strings (string characters or text) – Copy the Python code below into Main.py and Run.

```
# strings or text -- characters all strung together in a variable #
# strings should be encased in '' or "", just end with what you start #
number_of_friends = 10
number_of_friends = number_of_friends + 5
paycheck = 845.23
paycheck = 845
isWindowOpen = False
isDoorClosed = 7 > 4

last_name = 'Ryan'            # all text like a last name, with the
first character capitalized  #
passcode = "RyAN0884##"       # upper case, lower case characters,
numbers, and extended characters #
curr_grade = ''               # empty string with no characters #
blanks = "     "              # string of five blanks spaces #
number = '15'                 # string of numbers 15    #
# note:  number = '15' does not equal a value of 15  #

print(type(last_name))
print(last_name)
print(type(passcode))
print(passcode)
print(type(curr_grade))
print(curr_grade)
print(type(blanks))
print(blanks)
equality = (number == number_of_friends)
print(type(equality))
print(equality)
```



5) none (Null or Nill) – Copy the Python code below into Main.py and Run.

```
# None is the same as null - variable is void of value - no value #
# None string <> empty string #
# do not use None in comparisons or your Python program may crash #

number_of_friends = 10
number_of_friends = number_of_friends + 5
paycheck = 845.23
paycheck = 845
isWindowOpen = False
isDoorClosed = 7 > 4
last_name = 'Ryan'            # all text like a last name, with the
first character capitalized  #
passcode = "RyAN0884##"       # upper case, lower case characters,
numbers, and extended characters #
curr_grade = ''               # empty string with no characters #
blanks = "     "              # string of five blanks spaces #
number = '15'                 # string of numbers 15          #
# note:  number = '15' does not equal a value of 15  #

middle_name = None
print(type(middle_name))
print(middle_name)
```



6) For each of the variable types discussed in examples 1 to 5, create a variable for each variable type, assign the variable a value, use print() and print(type()) to display the value and variable type. Do not duplicate existing variable names or values from the screenshots above. The deliverables for this exercise is a screenshot of PyCharm after you run the Python code for each variable type, so you will have five screenshot deliverables.

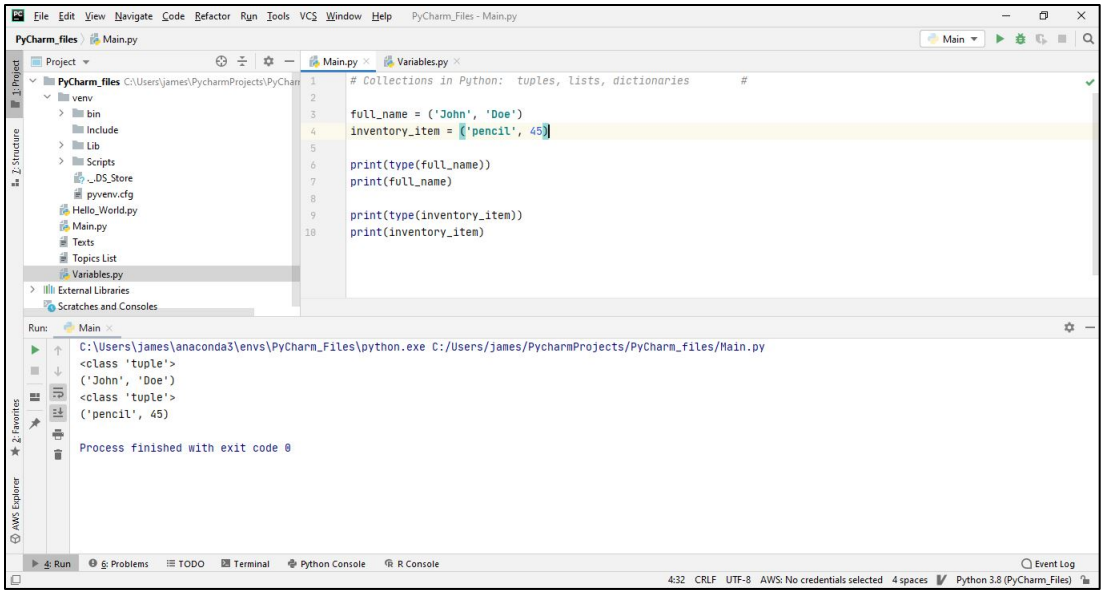
Using and Converting Variables:

1) Assignment operations: =, = not
2) Arithmetic operations: +, -, *, /, %, //,+=, -=, *=, /=, **
   a) % is modulus not percent: (a % b) or (5 % 2) = 1, Modulus returns the remainder of b divided into a.
   b) // is division floor: (a // b) or (5 // 2) = 2, Division floor returns the integer of b divided into a.
   c) += is equivalent to a = a + b or num_of_friends = num_of_friends + 2
   d) ** is power or a ** b or raise a to the power of b or val = 2 ** 2 = 4
      ** can be root, cube, etc. as a ** (1/b) or val = 2 ** (1/2) = 1.4142135623730951
3) Conditional operations: >, >=, <, <=, ==, !=
4) Variable conversion:
   int, float, or Boolean variable types can be converted to string
   Only numeric values can be converted to an int or float.
5) Exercise deliverables: Using variables from your earlier deliverables and new variables, provide PyCharm screenshots for each type of assignment, arithmetic, and conditional operations. Also, show variable conversions from int, float, and Boolean to string as well as string to int and float. You should have at least five screenshots (e.g., one for each category and extra screenshot for arithmetic operations to cover the remaining operands). Use the print() and print(type()) functions to show variable values and types.

Types of Collections in Python:

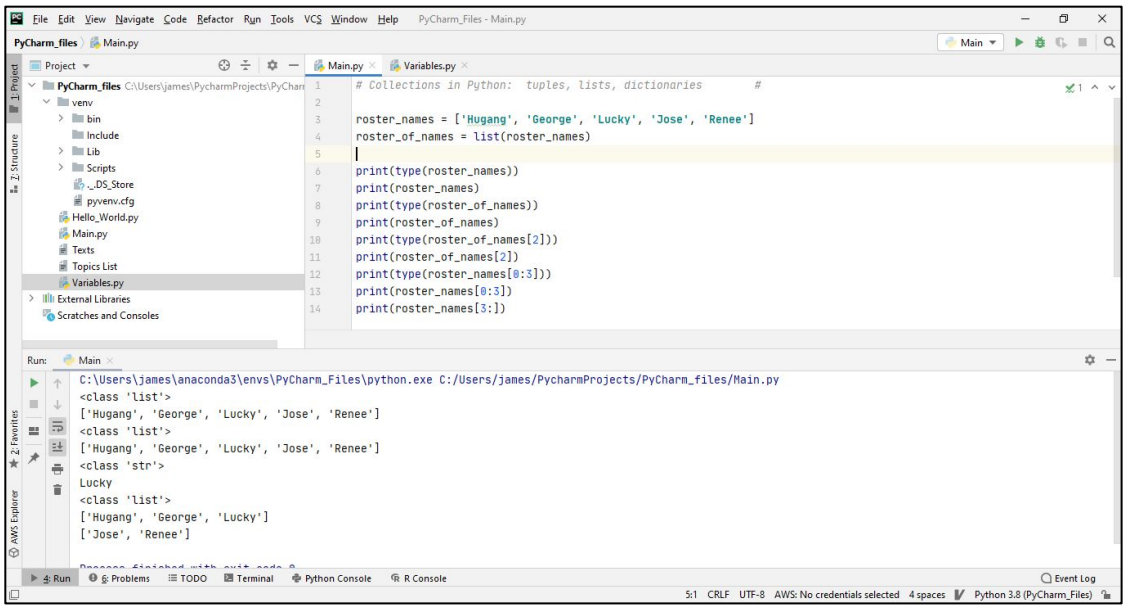
A collection in Python is a way to store multiple values in one variable location. The collections this tutorial reviews are: tuples, lists, and dictionaries.

1) ***tuple or tuples:***
   a) are similar to lists,
   b) tuples hold fewer values than lists,
   c) tuple values cannot be modified,
   d) tuple values are indexed sequentially with integers starting with the position 0,
   e) tuple values are comma separated,
   f) and tuple values ae surrounded by parentheses ().
   g) example: full_name = (‘first_name’, ‘last_name’) ≈ (‘John’, ‘Doe’)
      example: inventory_item = (‘item’, quantity) ≈ (‘pencil’, 45)



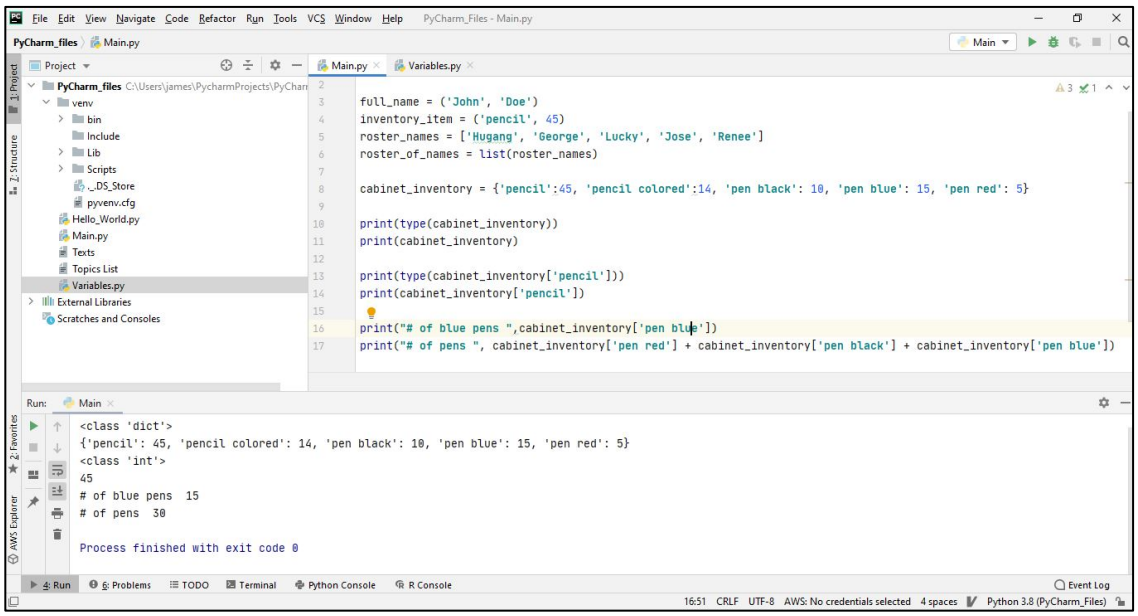
2) ***list or lists:***

a) are similar to tuples,
b) lists hold more values than tuples,
c) list values can be modified,
d) list values are indexed sequentially with integers starting with the position 0,
e) list values are comma separated,
f) and list values are surrounded by brackets [].
g) example: roster_names = [‘Hugang’, ‘George’, ‘Lucky’, ‘Jose’, ‘Renee’]



3) dictionary or dictionaries:

a) are different from tuples or lists,
b) each dictionary value is associated with a key,
c) dictionary values are mapped to keys,
d) dictionary values can be modified,
e) dictionary values are indexed by keys,
f) dictionary values are comma separated,
g) and dictionary values are surrounded by braces {}.
h) example: cabinet_inventory = {'pencil': 45, 'pen blue': 10, 'pen black': 15, 'pen red':5}



4) ***Exercise deliverables:*** Collect PyCharm screenshots for declaring tuple, list, and dictionary variables. Note in the screenshot above how the dictionary values were added. You should have at least three screenshots (e.g., one for each collection type variable). Use the print() and print(type()) functions to show collection variable types, values, and individual values.
Note: Attempting to access individual values by invalid indexes (e.g., integers higher than items represented in the tuple or list, or keys not included in a dictionary will result in a Python execution error.)

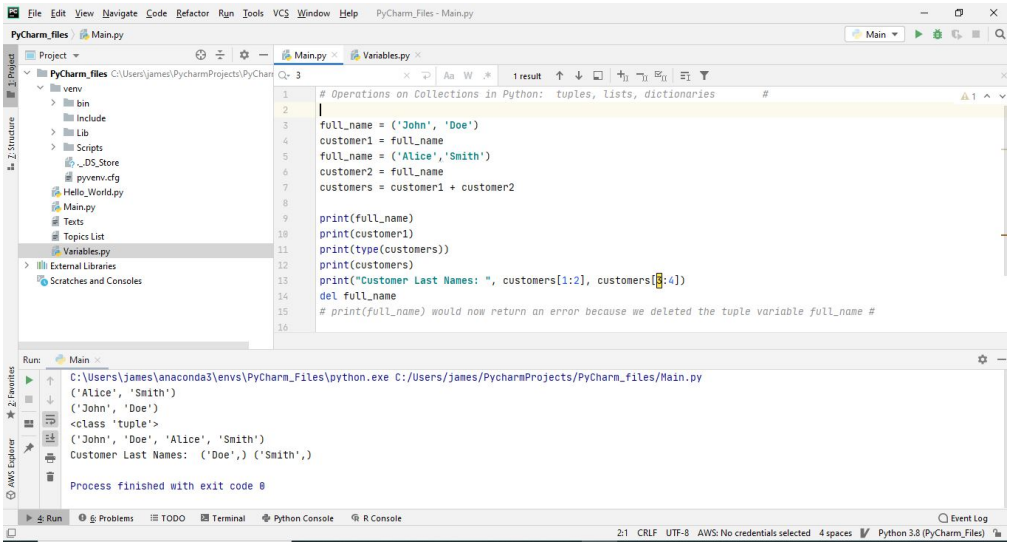
Collections Operations:

1) tuples:

a) tuple values cannot be modified, but can be reassigned, deleted, or combined with other tuples.

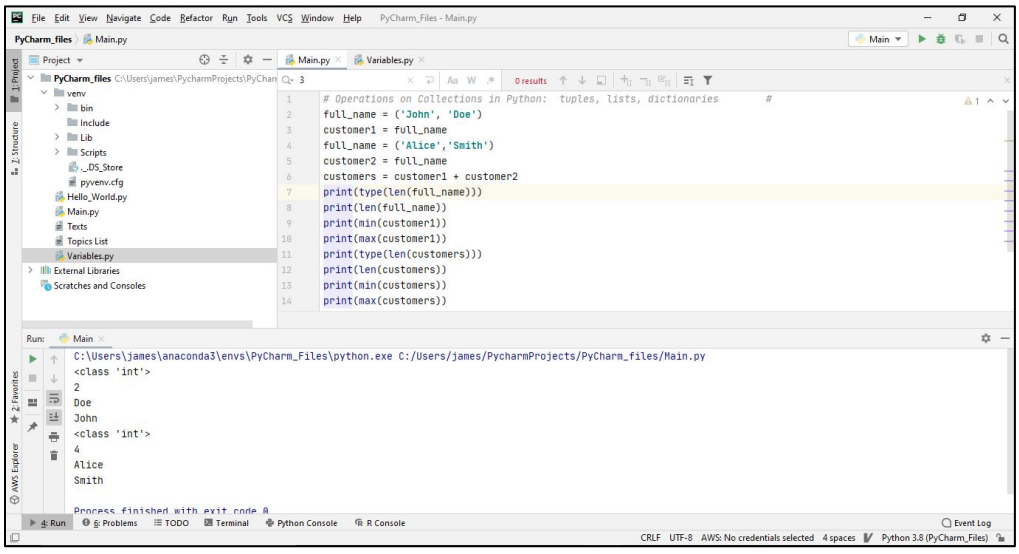
example: full_name = ('John', 'Doe') also customer 1 = full_name
full_name = ('Alice', 'Smith) also customer 2 = full_name

customer = customer1 + customer2



b) tuple values can be counted with the len() function, maximum values returned with the max() function, and minimum values returned with the min() function. For int or float, max() and min() return largest or smallest values, respectively. For string, max() and min() return strings based on alphabetical order (e.g., a being smallest and z being the largest).

example: print(len(full_name)), print(max(customers)), or print(min(customers))



2) lists:

a) list values can be reassigned, deleted, or combined as well as modified or cleared,

b) use del to delete a specific list value by its index. example: del roster_names[0]

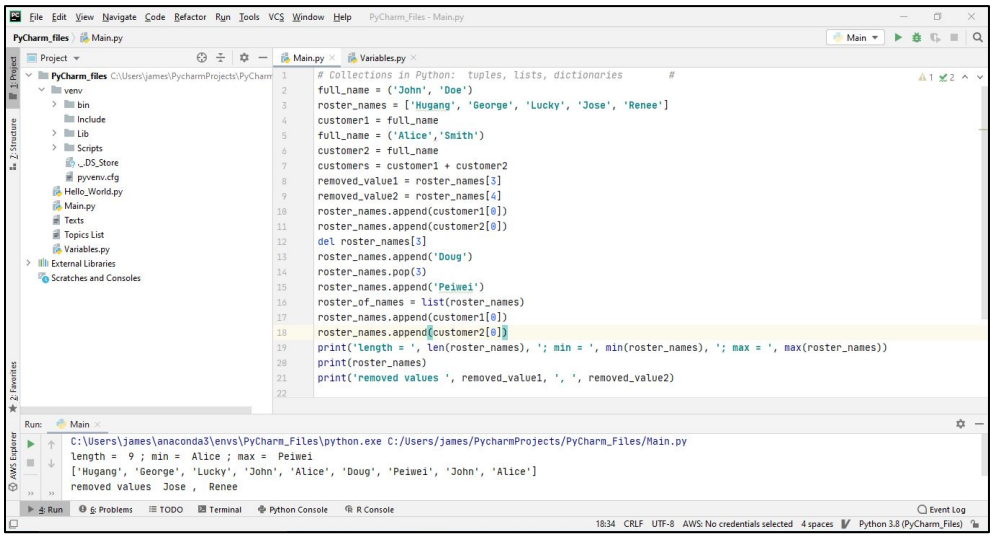
c) use del to delete an entire list of values. Example: del roster_names

d) use the list index to reassign a value in a list. example: roster_names[4] = 'Doug'

e) use the clear() function to clear all values in a list. example: roster_names.clear()

f) use the append() function to add a list value. example: roster_names.append('Doug')

g) use the len(), max(), and min() functions on list values. Example: PyCharm screenshot



3) dictionaries:

a) dictionary values can be reassigned, deleted, or combined as well as modified or cleared,

b) use del to delete a specific dictionary value by its key.
example: del cabinet_inventory['pencil colored']

c) use del to delete an entire dictionary of values. example: del cabinet_inventory

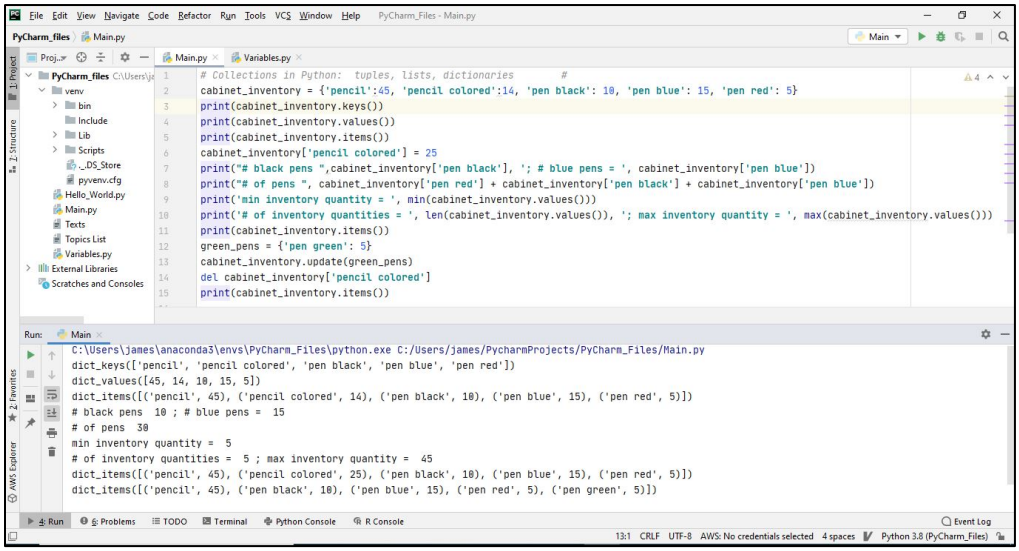
d) use a dictionary key to reassign a value. example: cabinet_inventory['pencil colored'] = 25

e) use the clear() function to clear all values in a dictionary.
example: cabinet_inventory.clear()

f) use the update() function to add a dictionary value.
example: cabinet_inventory.update({'pen green': 5})

g) use the len(), max(), and min() functions on dictionary values.

example: PyCharm screenshot



4) ***Exercise deliverables:*** Collect PyCharm screenshots for operations performed on tuple, list, and dictionary variables. Note in the previous screenshot how the dictionary values were added. You should have at least three screenshots (e.g., one for each collection type variable). Use the print() and print(type()) functions to show collection assignment and/or modification, cleared, deleted, and/or appended or updated. Note: Attempting to access individual values by invalid indexes (e.g., integers higher than items represented in the tuple or list, or keys not included in a dictionary will result in a Python execution error.)

Control Flow – if Statements:

1) ***"if" statements*** allow logic flow control in computer code.
   a. If a test is true, then specific code will be executed.
   b. If a test is false, then the specific code will be by-passed, an else if test will execute alternative code given the elif (i.e., else if) test is true.
   c. If the elif test is false a second alternative code can be executed.
   d. In Python, the syntax is:

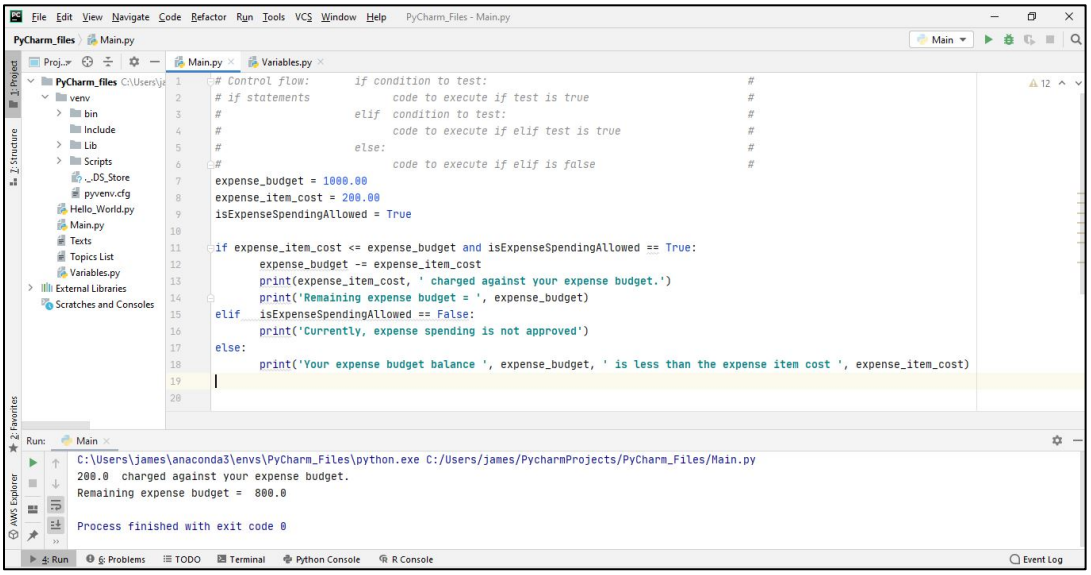
```
if test:
        execute code
elif test:
        execute code
else:
        execute code
```

Three outcomes based on two tests.

```
if test:
        execute code
else:
        execute code
```

Two outcomes based on one test.

   e. *Combining two or more test conditions with "and" requires all to be evaluated true.*
   f. *Combining two tests with "or" requires only one of the conditions to be evaluated true.*
   g. *"if" statements can be nested in "if" code, "elif" code, and/or "else" code.*
   h. *Including the "break" command stops the execution of the "if" statement.*
   i. "if", "elif", and "else" code is visible in the following PyCharm screenshot:



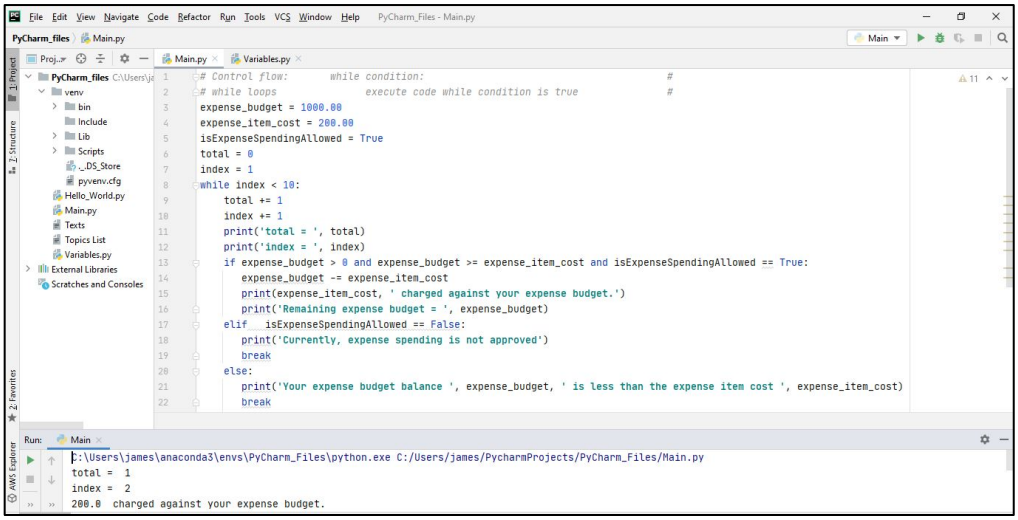
2) ***Exercise deliverables:*** Collect a PyCharm screenshot for Python code executing an "if" statement with one or more test conditions and resultant code, an "elif" test condition(s) and resultant alternative code, as well as "else" alternative code. Only one path of the "if", "elif", and "else" needs to be executed in the PyCharm screenshot, but the other two paths should be logically correct and executable.

Control Flow - While Loops and For Loops:

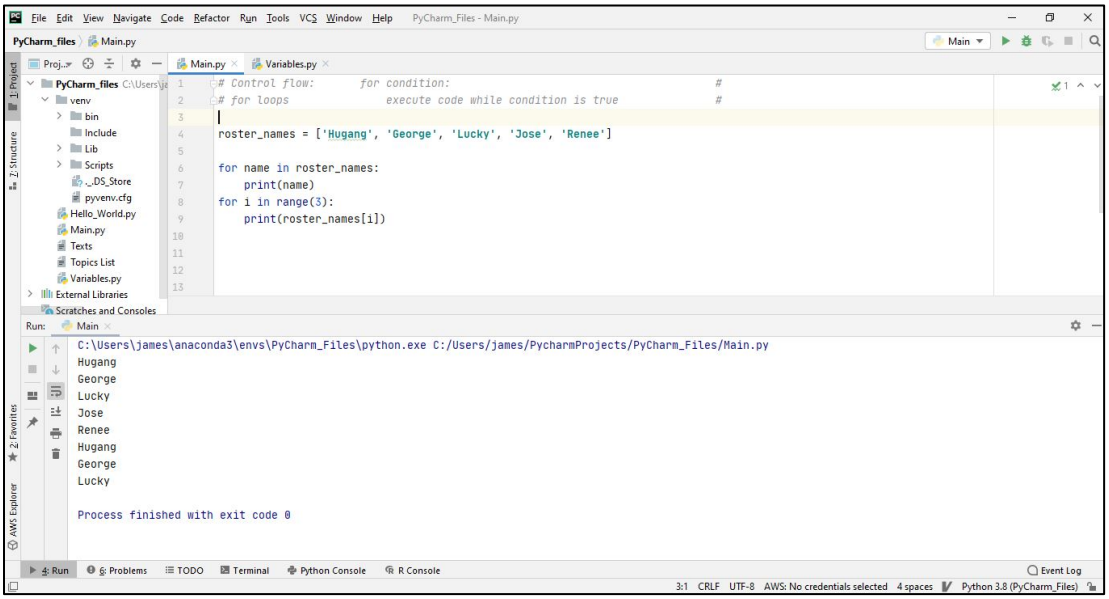
1) "while loops" allow execution of code while the test condition is true.
   a. Counters inside the while loop can make up the test condition.
   b. "if" statements and "for loops" may be nested inside "while loops"
   c. Caution: Including the "continue" command will force the "while loops" to continue even when the test condition is false.
   d. Including the "break" command will force the "while loops" to stop even when the test condition is true.
   e. "while loops" code to only look at even index values (2, 4, 6, 8)

```
total = 0
index = 1
while index < 10:
    if index % 2 == 0:
        index += 1
        continue
    total += index
    index += 1
```

   f. PyCharm screenshot of a "while loop" example. Enter the code in PyCharm and run the example.



2) "for loops" execute code in loops for a given condition
   a. Counters inside the for loop can make up the test condition.
   b. "if" statements and "while loops" may be nested inside "for loops"
   c. PyCharm screenshot of a "for loop". Enter the code in PyCharm and run the example.



3) ***Exercise deliverables:*** Collect two PyCharm screenshots for Python code executing "while loops" and "for loops".

Deliverables from Exercises Above:

Take the noted deliverables (i.e., PyCharm screenshots) from each exercise above (i.e., total of 20 screenshots), insert the screenshots into a MS Word document as you label each screenshot deliverable from the corresponding exercise, save the document as one PDF, and submit the PDF to the Tutorial Five Canvas Assignment uplink having the following qualities:

a. Your screenshot for each of the seven exercises above is included.
b. Each screenshot is labeled appropriately for each exercise.
   i. Becoming Familiar with PyCharm ..... 1 screenshot
   ii. Declaring Variables in Python ..... 5 screenshots
   iii. Using and Converting Variables ..... 5 screenshots
   iv. Types of Collections in Python ..... 3 screenshots
   v. Collections Operations ..... 3 screenshots

vi. Control Flow – “if” statements …….. 1 screenshot

vii. Control Flow – “while & for loops” ……. 2 screenshots

Total PyCharm screenshots to submit …….. 20 screenshots

c. All screenshots are legible output for each PyCharm exercise deliverable.