Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 2 Lab – 2a - 2e

```python
# ========================================================================
# LAB 2 Import and Headings
# ========================================================================
import math
# Docstrings lines for current exercises are minimized in screenshots and enclosed in double quotes, output still shown
# Functions to keep CONSTANTS and variables in local scope to not affect output of lab_2(a-e) code
# _Prefix before CONSTANTS for PyCharm and VSCode lowercase naming warnings


anguyen798
def lab_2_heading(letter: str = ""):
    """
    Enter the string value of letter encapsulated by quotes for the LAB 2 exercise
    :param letter: lab_2_heading(letter= **letter**:str)
    :return: None
    """
    print("*" * 100)
    print(f'\033[4mLAB 2{letter}\033[0m')  # f-string with underline f'\033[4m  \033[0m'
    print("-" * 50)

```

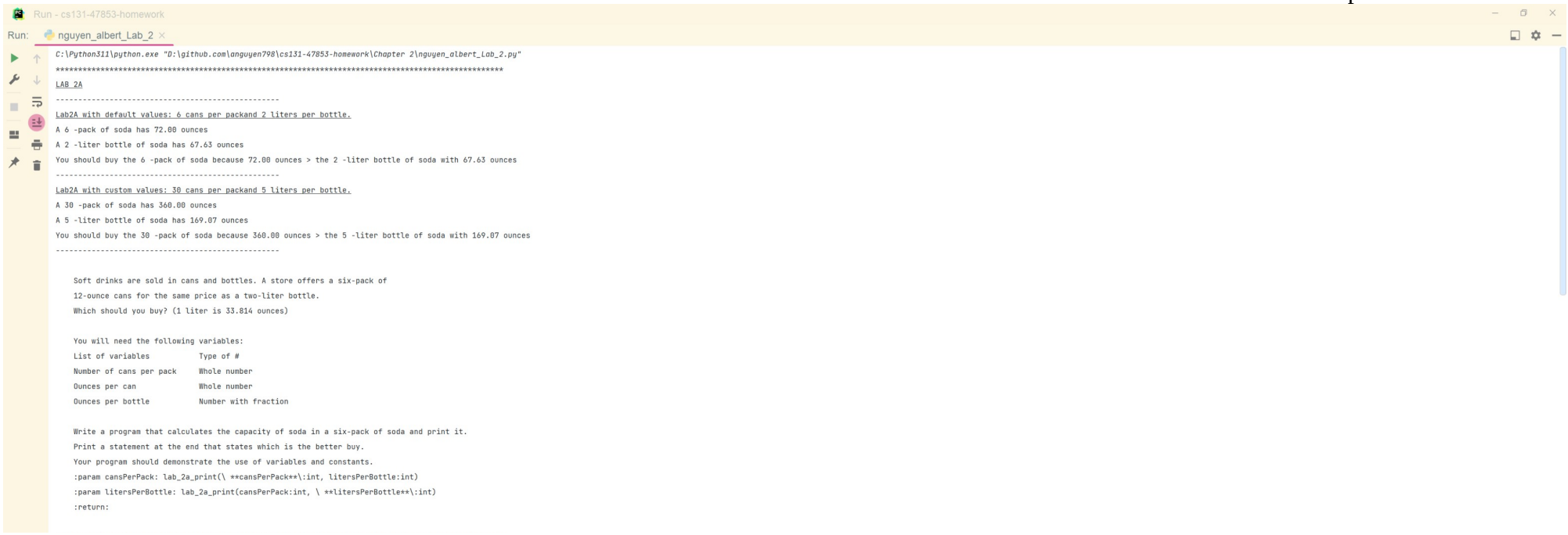
Lab2

Lab2 - Import and Heading function (adds separator, underlined heading and sub-separator); doc-strings included at end of output with exercise description and parameter details

```python
# ================================================================================
# LAB 2A
# ================================================================================


anguyen798 +1
def lab_2a_print(cansPerPack: int = 6, litersPerBottle: int = 2):
    r"""..."""
    _OUNCES_PER_CAN = int(12)
    _OUNCES_PER_PACK = cansPerPack * _OUNCES_PER_CAN
    _OUNCES_PER_LITER = 33.814
    _OUNCES_PER_BOTTLE = litersPerBottle * _OUNCES_PER_LITER

    if cansPerPack == 6 and litersPerBottle == 2:
        print(
            f'\033[4mLab2A with default values: {cansPerPack} cans per pack'
            f'and {litersPerBottle} liters per bottle.\033[0m')
    else:
        print(
            f'\033[4mLab2A with custom values: {cansPerPack} cans per pack'
            f'and {litersPerBottle} liters per bottle.\033[0m')
    print("A", cansPerPack, "-pack of soda has %.2f" % _OUNCES_PER_PACK, "ounces")
    print("A", litersPerBottle, "-liter bottle of soda has %.2f" % _OUNCES_PER_BOTTLE, "ounces")
    if _OUNCES_PER_PACK > _OUNCES_PER_BOTTLE:
        print("You should buy the", cansPerPack, "-pack of soda because %.2f" % _OUNCES_PER_PACK,
              "ounces > the", litersPerBottle, "-liter bottle of soda with %.2f" % _OUNCES_PER_BOTTLE, "ounces")
    elif _OUNCES_PER_PACK < _OUNCES_PER_BOTTLE:
        print("You should buy the", litersPerBottle, "-liter of soda because %.2f" % _OUNCES_PER_BOTTLE,
              "ounces > the", cansPerPack, "-pack of soda  with %.2f" % _OUNCES_PER_PACK, "ounces")
    print("-" * 50)


lab_2_heading(letter="A")
lab_2a_print()
lab_2a_print(30, 5)
print(lab_2a_print.__doc__)
```

A_Code

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 2 Lab – 2a - 2e

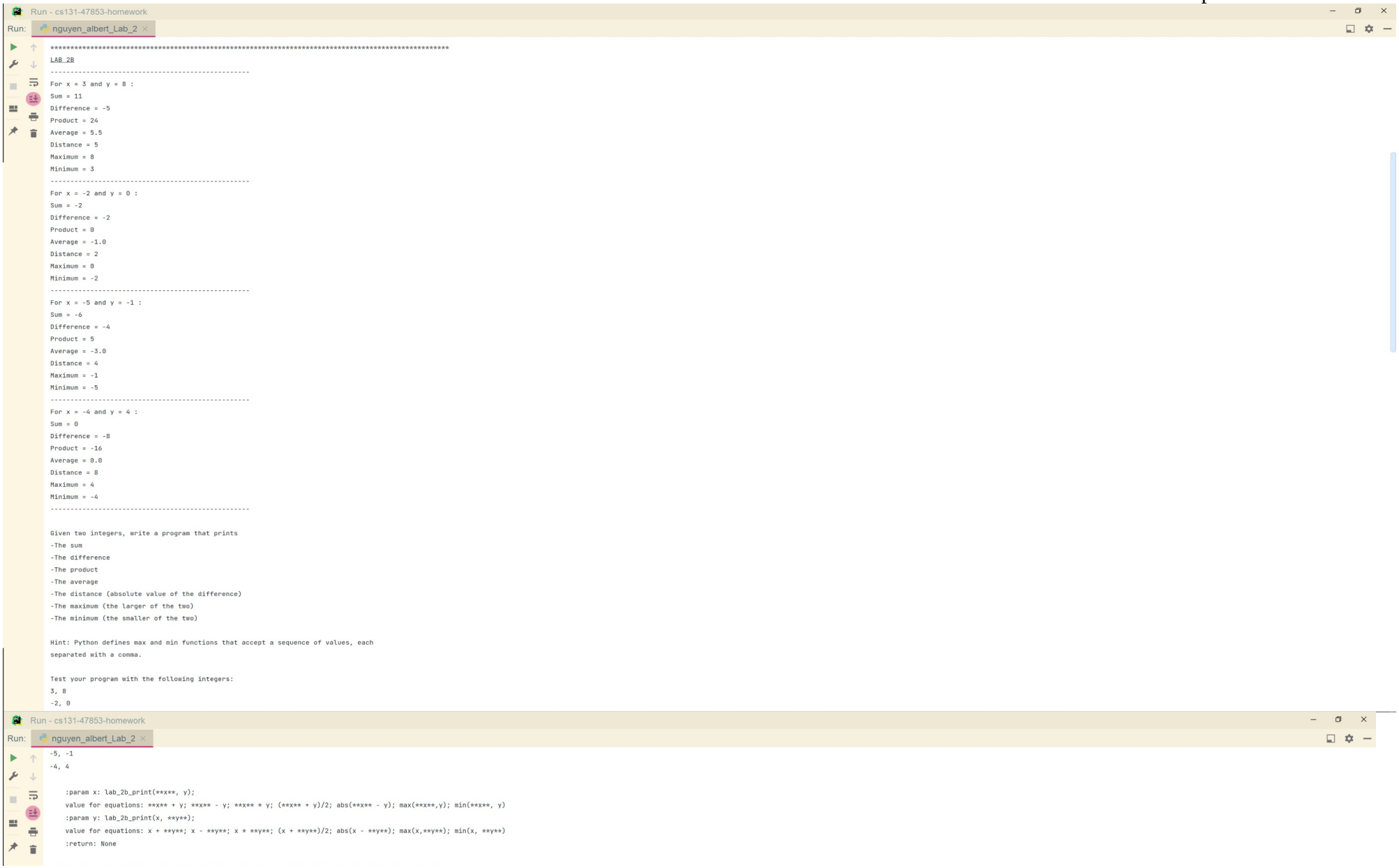


A_Console

```python
# ============================================================================
# LAB 2B
# ============================================================================


anguyen798 +1
def lab_2b_print(x: int, y: int):
    r"""..."""
    _LAB_2B_SUM = x + y
    _LAB_2B_DIFFERENCE = x - y
    _LAB_2B_PRODUCT = x * y
    _LAB_2B_AVERAGE = (x + y) / 2
    _LAB_2B_DISTANCE = abs(x - y)
    _LAB_2B_MAXIMUM = max(x, y)
    _LAB_2B_MINIMUM = min(x, y)
    print("For x =", x, "and y =", y, ":")
    print("Sum =", _LAB_2B_SUM)
    print("Difference =", _LAB_2B_DIFFERENCE)
    print("Product =", _LAB_2B_PRODUCT)
    print("Average =", _LAB_2B_AVERAGE)
    print("Distance =", _LAB_2B_DISTANCE)
    print("Maximum =", _LAB_2B_MAXIMUM)
    print("Minimum =", _LAB_2B_MINIMUM)
    print("-" * 50)


lab_2_heading(letter="B")
lab_2b_print(x=3, y=8)
lab_2b_print(x=-2, y=0)
lab_2b_print(x=-5, y=-1)
lab_2b_print(x=-4, y=4)
print(lab_2b_print.__doc__)
# lab_2b_print(x=int(input("Enter x = ")), y=int(input("Enter y = ")))  # manual input example
```

B_Code

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 2 Lab – 2a - 2e



B_Console

B_Console - Two variables(int) with multiple math formulas CONSTANTS

```python
# ========================================================================
# LAB 2C
# ========================================================================


anguyen798 +1
def lab_2c_print(x: int = 10, y: int = 6, a: int = 2, b: int = 8, c: int = 1):
    r"""..."""
    _LAB_2C_SQRT = math.sqrt(x + y)
    _LAB_2C_QUADRATIC = -b + math.sqrt(b ** 2 - 4 * a * c) / (2 * a)
    _LAB_2C_EXPONENT = x ** (y + 7)
    if x == 10 and y == 6 and a == 2 and b == 8 and c == 1:
        print(
            f'\033[4mLab2C with default values: x = {x}, y = {y}, a = {a}, b = {b}, c = {c}.\033[0m')
    else:
        print(
            f'\033[4mLab2C with custom values: x = {x}, y = {y}, a = {a}, b = {b}, c = {c}.\033[0m')
    print("For x =", x, ", y =", y, ". The square root of x + y =", _LAB_2C_SQRT,  # normal format
          "or %.2f" % _LAB_2C_SQRT)  # %f format
    print("For a =", a, ", b =", b, ", c =", c,
          ". The quadratic formula for (-b + sqrt(b^2 - 4ac)/2a =", _LAB_2C_QUADRATIC,
          "or %-10f" % _LAB_2C_QUADRATIC,  # %f format but with left-justify and without .2f to show un-rounded value
          "or", f'{_LAB_2C_QUADRATIC:.2f}')  # f-string format
    print("For x =", x, ", y =", y, ". The formula x ^ (y + 7) =", f'{_LAB_2C_EXPONENT:,}')  # f-string format
    print("-" * 50)


lab_2_heading(letter="C")
lab_2c_print()
lab_2c_print(2, 10, 10, 20, 4)
print(lab_2c_print.__doc__)
```

C_Code

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 2 Lab – 2a - 2e

C_Console

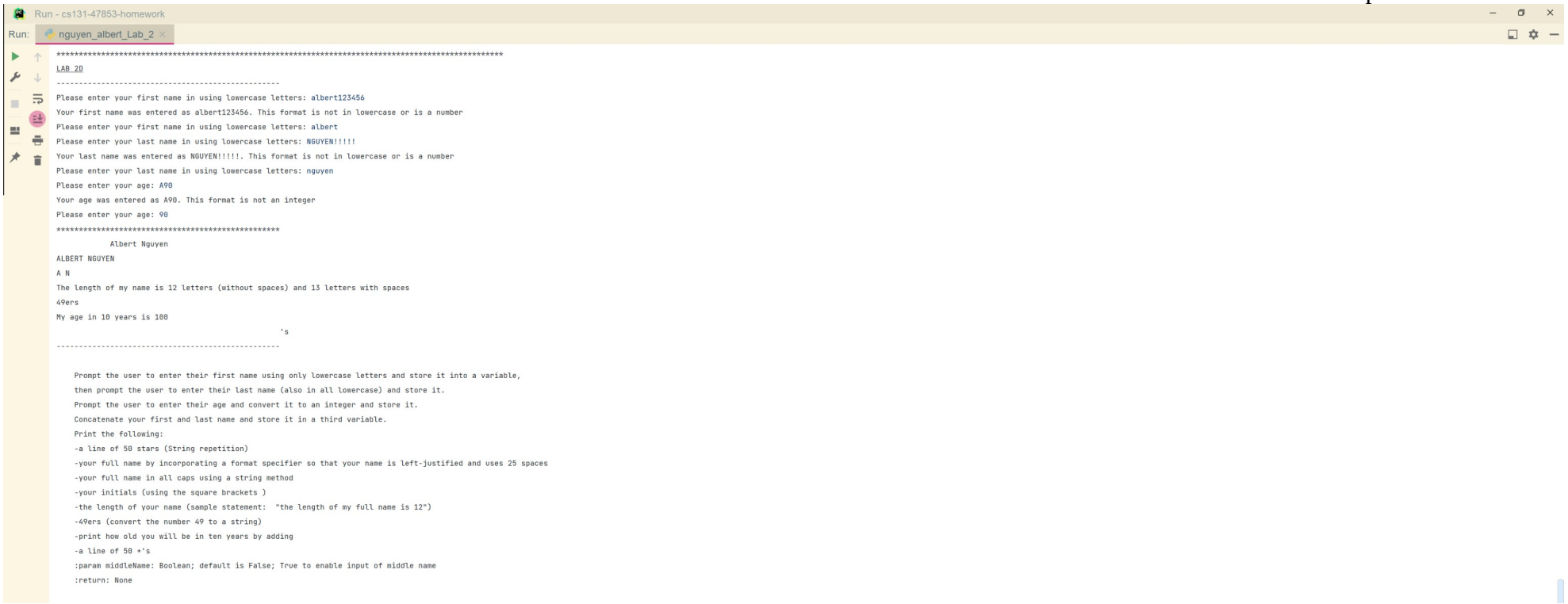
C_Console - Five variables(int) with three math formula CONSTANTS; passing function parameters into variables; import math

```
# ================================================================================
# LAB 2D
# ================================================================================


anguyen798 *
def lab_2d_print(middleName: bool = False):
    r"""..."""
    firstName = str(input("Please enter your first name in using lowercase letters: "))
    while firstName.islower() is False or firstName.isalpha() is False:  # OR conditional to check format
        print("Your first name was entered as %s. This format is not in lowercase or is a number" % firstName)
        firstName = input("Please enter your first name in using lowercase letters: ")
    if middleName:  # optional function parameter to enable middle name input, default is middleName=False
        middleName = input("Please enter your middle name in using lowercase letters: ")
        while middleName.islower() is False or middleName.isalpha() is False:
            print("Your middle name was entered as %s. This format is not in lowercase or is a number" % middleName)
            middleName = input("Please enter your middle name in using lowercase letters: ")
    lastName = input("Please enter your last name in using lowercase letters: ")
    while lastName.islower() is False or lastName.isalpha() is False:
        print("Your last name was entered as %s. This format is not in lowercase or is a number" % lastName)
        lastName = input("Please enter your last name in using lowercase letters: ")
    try:
        fullName = "%s %s %s" % (firstName.title(), middleName.title(), lastName.title())
        initials = "%s %s %s" % (firstName[0].upper(), middleName[0].upper(), lastName[0].upper())
    except AttributeError:
        fullName = "%s %s" % (firstName.title(), lastName.title())
        initials = "%s %s" % (firstName[0].upper(), lastName[0].upper())
        # initials = "{0}{1}".format(firstName[0].upper(), lastName[0].upper())  #using .format instead of %s specifier
    while True:
        try:  # allows input of invalid age in string format without crashing due to value error, loop until valid value
            ageString = input("Please enter your age: ")  # allows invalid value to be shown in ValueError print stmt
            age = int(ageString)
            if isinstance(age, int):  # break loop only if variable age is an integer
                break
        except ValueError:
            print("Your age was entered as %s. This format is not an integer" % ageString)
    print("*" * 50)
    print("%25s" % fullName.title())
    print(fullName.upper())
```

```
    print("%s" % initials)
    print("The length of my name is %s letters (without spaces) and %s letters with spaces"
          % (len(fullName.replace(" ", "")), len(fullName)))  # 2nd line due to PEP8 over character limit
    # print(f'The length of my full name is {len(fullName)}')  # length of name in f-string format
    print("%sers" % 49)
    # print("{0}ers".format(49))  # using .format instead of %s specifier
    print("My age in 10 years is %s" % (age + 10))
    if age > 122:
        print("Your age is %s. Are you are a vampire?" % age)
    print(" " * 50 + "'s")


lab_2_heading(letter="D")
lab_2d_print()
print("-" * 50)
print(lab_2d_print.__doc__)
```

D_Code

Albert Nguyen
CS131-47853 MW 06:00 PM – 08:30 PM
Chapter 2 Lab – 2a - 2e

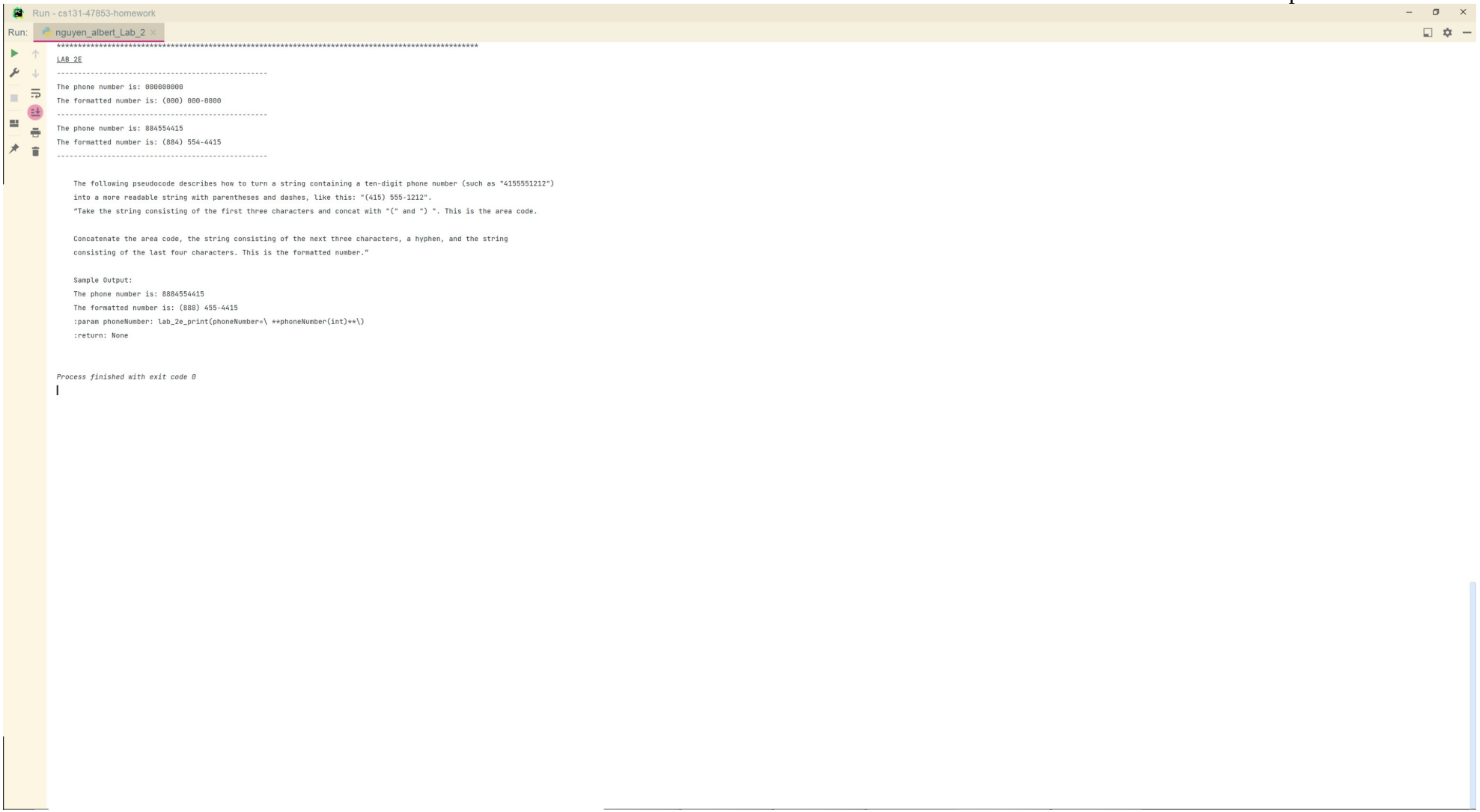


D_Console

D_Console - Prompts, arrays, length function, string formatting; with while loop and exceptions to force user to input correct type

```
# ========================================================================
# LAB 2E
# ========================================================================



anguyen798 +1
def lab_2e_print(phoneNumber=0000000000):
    """
    The following pseudocode describes how to turn a string containing a ten-digit phone number (such as "4155551212")
    into a more readable string with parentheses and dashes, like this: "(415) 555-1212".
    “Take the string consisting of the first three characters and concat with "(" and ") ". This is the area code.

    Concatenate the area code, the string consisting of the next three characters, a hyphen, and the string
    consisting of the last four characters. This is the formatted number.”

    Sample Output:
    The phone number is: 8884554415
    The formatted number is: (888) 455-4415
    :param phoneNumber: lab_2e_print(phoneNumber=\ **phoneNumber(int)**\)
    :return: None
    """
    phoneNumber = "%09.f" % phoneNumber
    phoneNumberFormatted = "(" + phoneNumber[:3] + ") " + phoneNumber[3:6] + "-" + phoneNumber[-4:]
    # start at array[0] to before array[3], start at array[3] to before array[6], start at array[len(variable) -4 = 9-4]
    # phoneNumberFormatted = f'({phoneNumber[:3]}) {phoneNumber[3:6]}-{phoneNumber[6:]}'  # f-string code
    print("The phone number is: " + phoneNumber)
    print("The formatted number is: " + phoneNumberFormatted)
    print("-" * 50)



lab_2_heading(letter="E")
lab_2e_print()
lab_2e_print(phoneNumber=884554415)
print(lab_2e_print.__doc__)
```

E_Code


Run - cs131-47853-homework
Run:
nguyen_albert_Lab_2
****
LAB 2E

The phone number is: 000000000
The formatted number is: (000) 000-0000

The phone number is: 884554415
The formatted number is: (884) 554-4415

The following pseudocode describes how to turn a string containing a ten-digit phone number (such as "4155551212")
into a more readable string with parentheses and dashes, like this: "(415) 555-1212".
“Take the string consisting of the first three characters and concat with "(" and ") ". This is the area code.
Concatenate the area code, the string consisting of the next three characters, a hyphen, and the string
consisting of the last four characters. This is the formatted number.”
Sample Output:
The phone number is: 8884554415
The formatted number is: (888) 455-4415
:param phoneNumber: lab_2e_print(phoneNumber=\ **phoneNumber(int)**\)
:return: None
Process finished with exit code 0

E_Console