Lab 04: Functions

Goals for this lab:

1. Define a function
2. Pass data to a function
3. Return data from a function
4. Design and implement a Python program that has multiple .py files
5. Use docstrings to provide appropriate documentation for functions
6. Use the PyTest module to construct unit tests for individual functions
7. Write and execute simple Python programs using the given instructions

General Guidelines:

- Use meaningful variable names
- Use appropriate indentation
- Use comments, especially for the header, variables, and blocks of code.
- Use docstrings for functions

Example Header:

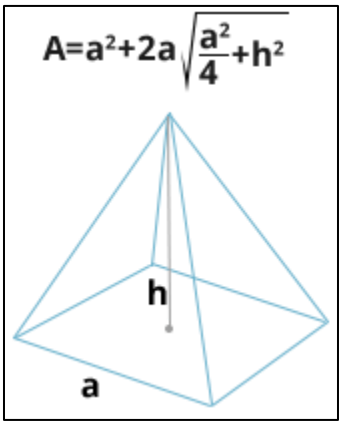
```
# Author: Your name
# Date: Today's date
# Description: A small description in your own words that describes what
the program does. Any additional information required for the program to
execute should also be provided.
```

Example Function Docstring:

```
"""
General function description
:param p1: DESCRIPTION
:type p1: TYPE
:param p2: DESCRIPTION
:type p2: TYPE
:return: DESCRIPTION.
"""
```

1. Surface Area

The surface area of a square pyramid can be described as follows:

2. 

This means that the surface area is composed of the base area (i.e., the area of bottom square) plus the side area (i.e., the sum of the areas of all four triangles). You are given the following incomplete code in **pyradmidArea.py** to calculate and print out the total surface area of a square pyramid:

```
# Author: Your name
# Date: Today's date
# Description: A small description in your own words that describes what the program does.
# Any additional information required for the program to execute should also be provided.

# function definitions
def calcBaseArea(side):
    return side ** 2

# add your function definition for calcSideArea here

# add your function definition for prntSurfArea here

def main():
    side = float(input("Enter the side length of the base of the square pyramid in feet: "))

    height = float(input("Enter the height of the square pyramid in feet: "))

    base_area = calcBaseArea(side)
    print(f"Base surface area of the square pyramid is {base_area} square feet.")

    # add your function to calculate the side area and assign
    # the result to side_area, then print the result

    # add your function call to print the total surface area


if __name__ == "__main__":
    main()
```

This program currently prompts for and reads in the side length of the base (side) and height of a square pyramid (height) in feet, and then calculates the surface area of the base. To complete this lab, you will need to define a function named `calcSideArea` which takes in the side length and height of a pyramid as parameters, and then calculates and returns the total side area of a square pyramid. Also, make sure you output the resulting side area with an appropriate message.

Next, define a function named `prntSurfArea` that takes in the total base area and total side area as parameters and prints out the total surface area of the square pyramid inside this function. This function will not return any data.

Make sure you save your **pyradmidArea.py** as you will be submitting this file for Lab 04.

2. ABT (Always Be Testing)

Now that you have constructed your program to calculate the area of a pyramid, let's use PyTest to setup test cases to ensure that it works. Create a new file named **test_pyramidArea.py** (it must have test_ at the front for PyTest to automatically discover it), and import the `pytest` library and your `pyramidArea` file. Then construct the following tests for the following test cases (don't forget to start each test function with `test_`):

- Test `calcBaseArea()` by passing it the number 15 and asserting that the result will be 225
- Test `calcBaseArea()` by passing in the string "5" and use a mark to indicate that it will fail and that the reason is because the input should not be text
- Test `calcSideArea()` by passing in 15 for the side parameter, 5 for the height parameter, and ensuring the resulting value is between 270.41 and 270.42
- Test `calcSideArea()` by passing in 10 for the side parameter, 3 for the height parameter, and ensuring the result, when rounded to two decimal points of precision, is equal to 116.62
- Test `prntSurfArea()` by passing in 15 for the side parameter, 10 for the height parameter, and use a mark to skip this test with the reason being that the function only prints text to the screen

Next, execute your tests and be sure that three of them passed, 1 of them was skipped, and 1 of them xfailed.

Make sure you save your **test_pyramidArea.py** as you will be submitting this file for Lab 04.

3. Painting

Being rather creative, some people determined that the various letters, numbers, and symbols we type into a computer can be used for the creation of works of art. ASCII art, as it is sometimes called, can be large and complex covering many lines and with hundreds or thousands of symbols or as short as a single line and a few characters, who knew ¯_(ツ)_/¯

For this program, **painter.py**, you will be creating a program that will allow the user to choose from four options as to what ASCII art painting they would like to see. They are also allowed to choose a single symbol border for the painting. To facilitate this, you will need to create a number of functions, in addition to a `main()` function, that coordinates the activities of painting the artwork.

Additionally, while two ASCII art images have been provided to you, **sailingShipt.txt** and **sleepingCat.txt** , you will need to select two additional ASCII art images to output as well. The website ASCII Art Archive , has a wide range of art for you to choose from. You can of course choose what you would like so long as it is not offensive or inappropriate. Also, be careful if your image contains \ as that is the escape character, and you will need to escape it with another \ (so you should have \\ to have the correct output). You will need to copy all four of these pieces of art into their appropriate functions so they can be output.

Your program should contain the following functions:

- A function named `intro()` that takes in no parameters and returns an integer representing the user's choice of art and a string representing the user's desired border, and
    - Welcomes the user to the program
    - Provides them with a numeric list of ASCII art options
    - Prompts for and stores their choice of art
    - Prompts for and stores the symbol they would like to use for the border
    - Returns the user's information
- A function named `printHeaderFooter()` that takes in a border and a size as parameters, returns nothing, and outputs the border size number of times on the same line
- A function named `sleepingCat()` that takes in a border as a parameter, returns nothing, and
    - Stores the image of the cat in one or more variables
    - Calls the `printHeaderFooter()` function and provides it the border and an appropriate size to completely cover the top of the painting
    - Outputs each line of the ASCII art with the border symbol before and after the line. Note, the right side of the border should be a straight line up and down so think about how you can reserve enough white space to ensure this happens
    - Calls the `printHeaderFooter()` function and provides it the border and an appropriate size to completely cover the bottom of the painting
- A function named `sailingShip()` that takes in a border as a parameter, returns nothing, and
    - Stores the image of the ship in one or more variables
    - Calls the `printHeaderFooter()` function and provides it the border and an appropriate size to completely cover the top of the painting
    - Outputs each line of the ASCII art with the border symbol before and after the line. Note, the right side of the border should be a straight line up and down so think about how you can reserve enough white space to ensure this happens
    - Calls the `printHeaderFooter()` function and provides it the border and an appropriate size to completely cover the bottom of the painting
- Two additional functions containing your ASCII art choices that each take in a border as a parameter, return nothing, and follow the same steps for the painting border
- A function named `blank()` that takes in a border as a parameter, returns nothing, and
    - Calls the `printHeaderFooter()` function and provides it the border and an appropriate size to completely cover the top of the painting
    - Outputs five lines of five spaces with the border in front of and after each line. This represents a blank canvas
    - Calls the `printHeaderFooter()` function and provides it the border and an appropriate size to completely cover the bottom of the painting
- A function named `main()` that takes in no parameters, returns nothing, and
    - Calls the intro function and stores the results in appropriate variables
    - Using `if`/`elif`/`else` statements and the user's choice of art, call the appropriate art painting function and provide it the user selected border

        - If the user enters a value that is not a menu option, call the blank function, pass it the border, inform the user that you do not seem to have that painting, and use the `exit(-1)` function to immediately terminate the program
    - Tell the user that you hope they enjoyed their art

Also, since you now have a `main()` function, do not forget to call it at the end of your program, or you may be a bit frustrated when nothing happens (ノ°□°)ノ ⌒ ┻━┻

Make sure you save your **painter.py** as you will be submitting this file for Lab 04.

4. Painting With a Twist

Now that you have completed your **painter.py** program, you may notice that it is starting to become rather large! To remedy this, you will need to create a new file named **painterMain.py** that contains only your `main()` function, and a **painterFuncs.py** file that contains all of your other functions.

You should then perform the following updates:

- Create docstrings for each of your functions in **painterFuncs.py** which provides a brief description of the function and the names and purpose of each parameter
    - Remember to add the docstring immediately under the function header
- Include the **painterFuncs.py** module in **painterMain.py**
- Update all of the function calls in **painterMain.py** so they know which module they came from

Once you've done this, double check your work to ensure the program still provides all of the correct output.

Make sure you save your **painterMain.py** and **painterFuncs.py** as you will be submitting these files for Lab 04.

Submission

Once you have completed this lab it is time to turn in your work. Please submit the following files to the Lab 04 assignment in Canvas:

- **pyradmidArea.py**
- **test_pyramidArea.py**
- **painter.py**
- **painterMain.py, painterFuncs.py**

Sample Output

pyradmidArea.py

```
Enter the side length of the base of the square pyramid in feet: 15
Enter the height of the square pyramid in feet: 5
Base surface area of the square pyramid is 225.0 square feet.
Total surface area of all four sides of the square pyramid is
270.41634565979916 square feet.
The total surface area of the square pyramid is  495.41634565979916
square feet.
```

test_pyradmidArea.py

```
==================== test session starts ======================
collecting ... collected 5 items

test_pyramidArea.py::test_calcBaseArea PASSED                        [ 20%]
test_pyramidArea.py::test_calcBaseAreaText XFAIL (Input should not be
text!)                                                               [ 40%]
@pytest.mark.xfail(reason="Input should not be text!")
    def test_calcBaseAreaText():
>       pa.calcBaseArea("5")

test_pyramidArea.py:10:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

side = '5'

    def calcBaseArea(side):
>       return side ** 2
E       TypeError: unsupported operand type(s) for ** or pow(): 'str' and
'int'

pyramidArea.py:11: TypeError

test_pyramidArea.py::test_calcSideAreaBetween PASSED                 [ 60%]
test_pyramidArea.py::test_calcSideAreaRound PASSED                   [ 80%]
test_pyramidArea.py::test_prntSurfArea SKIPPED (Only prints text to the
screen.)                                                             [100%]
Skipped: Only prints text to the screen.


============= 3 passed, 1 skipped, 1 xfailed in 0.08s =============
```

painter.py

```
Welcome to the painting printer!
    We have many options:
    1. The S.S. Satisfaction
    2. Mina in Repose
    3. [Insert Painting]
    4. [Insert Painting]
Please select a painting to print: 1
What border would you like around your painting: *
**********************************
*       |    |    |               *
*      )_)  )_)  )_)              *
*     )___))___))___)\            *
*    )____)____)_____)\\          *
*  _____|____|____|____\\\__      *
*  \    Satisfaction   /          *
* ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  *
**********************************
We hope you enjoy your art!

Welcome to the painting printer!
    We have many options:
    1. The S.S. Satisfaction
    2. Mina in Repose
    3. [Insert Painting]
    4. [Insert Painting]
Please select a painting to print: 2
What border would you like around your painting: #
##################################
#       |\      _,,,---,,_       #
# ZZZzz /,`.-'`'    -.  ;-;;,_   #
#      |,4-  ) )-,_. ,\ (  `'-' #
#     '---''(_/--'  `-'\_)       #
##################################
We hope you enjoy your art!
```

```
Welcome to the painting printer!
    We have many options:
    1. The S.S. Satisfaction
    2. Mina in Repose
    3. [Insert Painting]
    4. [Insert Painting]
Please select a painting to print: 6
What border would you like around your painting: @
@@@@@@@@@
@       @
@       @
@       @
@       @
@       @
@@@@@@@@@
Hmmmm....we don't seem to have that painting.
```

painterMain.py, painterFuncs.py

Same as for **painter.py**

Rubric

For each program in this lab, you are expected to make a good faith effort on all requested functionality. Each submitted program will receive a score of either 100, 75, 50, or 0 out of 100 based upon how much of the program you attempted, regardless of whether the final output is correct (See table below). Additionally, it is expected that you will provide a header comment at the beginning of each program, and so 10 points will be deducted from each program that is missing the header comment. The scores for all of your submitted programs for this lab will be averaged together to calculate your grade for this lab. Keep in mind that labs are practice both for the exams in this course and for coding professionally, so make sure you take the assignments seriously.

Score	Attempted Functionality
100	100% of the functionality was attempted
75	<100% and >=75% of the functionality was attempted
50	<75% and >=50% of the functionality was attempted
0	<50% of the functionality was attempted