Summative Assessment 1

By Antonio III

Chapters

Chapter 1

Chapter 2

Chapter 3

Ex01

Ex02

Ex03

Ex04

Chapter 4

Ex01

Ex02

Ex03

Ex04

Ex05

Chapter 5

Ex01

Ex02

Ex03

Ex04

Ex05

Ex06

Chapter 6

Ex01

Ex02

Ex03

Ex04

Ex05

back

Ch1-ex01

```
name = input("Hello, user!\nWhat is your name? \n") #no whitespace between \n and "What" because it creates a blank space in the output
age = int(input("What is your age? \n")) #you can use \n like this here because there is no letter after the \n that messes up the formattin
name_len = len(name) #to store the length of the name
age_one_year = age + 1 #used a variable to store so that all print functions use f-string formatting

print(f"It is good to meet you, {name.title()}!") #used .title() method to capitalize the first letters of the name
print(f"The length of your name is: \n{name_len}")
print(f"You will be {age_one_year} in a year.")
```

```
Hello, user!
What is your name?
antonio
What is your age?
20
It is good to meet you, Antonio!
The length of your name is:
7
You will be 21 in a year.
```

back

Ch1-ex02

```
first_num = int(input("Enter a number \n"))
second_num = int(input("Enter a second number \n"))

sum = first_num + second_num
diff = first_num - second_num
prod = first_num * second_num
quo = first_num / second_num
rem = first_num % second_num
print(f"The sum of the numbers is: {sum}!")
print(f"The difference of the numbers is: {diff}!")
print(f"The product of the numbers is: {prod}!")
print(f"The quotient of the numbers is: {quo}!")
print(f"The remainder of the numbers is: {rem}!")
```

```
\Chapter 1-Python Refresher\Exercise 02 - Ma
Enter a number
20
Enter a second number
50
The sum of the numbers is: 70!
The difference of the numbers is: -30!
The product of the numbers is: 1000!
The quotient of the numbers is: 0.4!
The remainder of the numbers is: 20!
```

back

Ch1-ex03 code

```
entered_triangle_sides = 0 #A counter to see how many sides have been input
triangle_sides = [] #Created a list to store all the sides that have been input

while entered_triangle_sides <3: #Runs a loop to run the code below repeatedly
    triangle_side = int(input("Enter a triangle side:\n"))
    triangle_sides.append(triangle_side) #Adds the input number to the list
    entered_triangle_sides = entered_triangle_sides +1 #Once the code above has been executed, this will add a +1 to the variable on line 6


#Checks if the input sides pass the triangle inequality theorem
if triangle_sides[0] + triangle_sides[1] > triangle_sides[2] and triangle_sides[1] + triangle_sides[2] > triangle_sides[0] and triangle_sides[2] + triangle_sides[0] > triangle_sides[1]:

    print("You have entered a triangle! Congratulations!") #The input sides must pass the triangle inequality theorem for this to execute

    #This is for the extension problem. This executes if the condition (line 16) is met
    if triangle_sides[0] == triangle_sides[1] and triangle_sides[1] == triangle_sides[2]:
        print("And you entered an Equilateral Triangle!")

    elif triangle_sides[0] == triangle_sides[1] or triangle_sides[1] == triangle_sides[2] or triangle_sides[2] == triangle_sides[0]:
        print("And you entered an Isosceles Triangle!")

    else:
        print("And you have entered a Scalene Triangle!")

else:
    print("This is not a valid triangle! :((\nSad!")
```

back

Ch1-ex03 output

```
Enter a triangle side:
2
Enter a triangle side:
3
Enter a triangle side:
4
You have entered a triangle! Congratulations!
And you have entered a Scalene Triangle!
```

back

Ch1-ex04 code

```
three_nums = {"first":None,"second":None,"third":None} # Initialize a dict that will collect the 3 numbers entered

for n in three_nums.keys(): # This for loop iterates as much as the amount of items in the list (3)
    entered_num = int(input(f"Input the {n} number: \n")) # Turns the input into an integer and store it into the variable
    three_nums[n] = entered_num # Add the entered nunmbers as the keys' value

# Sort the dictionary by their values-pair in descending order by using a custom sorting key. This returns a list-type so we use di

# The lambda function is how we're going to sort the dictionary. In our case, we wrote 'item[1]' which sorts it by the second eleme

#If we don't retrieve the .items(), and we remove the conversion to a dict (it causes errors), this returns a list (from sorted())

sorted_nums = dict(sorted(three_nums.items(), key = lambda item: item[1], reverse = True))

# Store the variable keys and values to these variables. 'desc' means 'descending'
desc_num_keys = list(sorted_nums.keys())
desc_num_values = list(sorted_nums.values())

# Find the greatest number in the list using multiple if-else statements. This automatically finds the the highest entered number,
if desc_num_values[0] > desc_num_values[1] and desc_num_values[0] > desc_num_values[2]: # Both conditions must be true for the code
    print(f"The {desc_num_keys[0]} number is the greatest, being {desc_num_values[0]}.") # Outputs the greatest number as well as t

# In case of all numbers being equal or that there are 2 greatest numbers
else:
    # All numbers are equal
    if desc_num_values[0] == desc_num_values[1] and desc_num_values[0] == desc_num_values[2]:
      print(f"All numbers are equal, that being {desc_num_values[0]}.")

    # There are 2 greatest numbers
    else:
        print(f"The {desc_num_keys[0]} and {desc_num_keys[1]} numbers are both the greatest, being {desc_num_values[0]}.")
```

back

Ch1-ex-04 output

```
Input the first number:
20
Input the second number:
50
Input the third number:
30
The second number is the greatest, being 50.
```

back

Ch1-ex05 code

```
iterations = 0

# Initialize a boolean to track if the user wants to exit the loop
exit_loop = False

# Write the 'while loop' code block
while exit_loop == False:

  # Initialize a boolean to be used later in the code for another 'while loop'. It is placed inside the loop so its variable would reset per iteratio
  invalid_char = True


  if iterations == 1:
    # So the statement is grammmatically correct if we've looped for exactly one (1) time
    enter_char = input(f"You are in a 'while loop'. You have looped {iterations} time. Would you like to loop another iteration? (y/n) \n")
  else:
    enter_char = input(f"You are in a 'while loop'. You have looped {iterations} times. Would you like to loop another iteration? (y/n) \n")

  # I created another 'while loop' and placed it inside the main loop because I want to have a different outcome when the user enters a different cha

  while invalid_char != False: #This is set up to pass the condition of the boolean (in line 13) so we have to go through this loop at least once, fo
    if enter_char == 'y':
      invalid_char = False # We replaced the boolean value here because we need to fail the condition of this second loop, so we can continue to the
      # Removed 'continue' statement in this line because even without it, once the last code has been executed (line 16), we will automatically go t

      iterations +=1 # We add integer 1 to the iterations count since this marks the completion of 1 loop

    elif enter_char == 'n':
      exit_loop = True # Set boolean to fail the condition of the main loop
      break # Exit the loop we're currently in (the second 'while loop')

    else:
      # Initialize a variable that takes an input to see if the respone will be a valid character, else this code will keep executing. We don't need
      enter_char = input(f"Invalid input. Would you like to loop another iteration? (y/n) \n")
```

back

Ch1-ex05 output

```
You are in a 'while loop'. You have looped 0 times. Would you like to loop another iteration? (y/n)
y
You are in a 'while loop'. You have looped 1 time. Would you like to loop another iteration? (y/n)
y
You are in a 'while loop'. You have looped 2 times. Would you like to loop another iteration? (y/n)
1
Invalid input. Would you like to loop another iteration? (y/n)
n
```

back

Ch1-ex06

```
for num in range (1, 101):

  # We use modulo division to check if the '
  # We put this condition first because we r
  if num % 3 == 0 and num % 5 == 0:
    print("FizzBuzz")

  elif num % 3 == 0:
    print("Fizz")

  elif num % 5 == 0:
    print("Buzz")

  # If the iteration fails all the above cor
  else:
    print(num)
```

```
Buzz
56
Fizz
58
59
FizzBuzz
61
62
Fizz
64
Buzz
Fizz
67
68
Fizz
Buzz
71
Fizz
73
74
FizzBuzz
76
77
Fizz
79
Buzz
Fizz
82
83
Fizz
Buzz
86
Fizz
88
89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

back

Ch1-ex07

```
for num in range(1,101):
  #if the current value of num, when divided
  if num % 2 == 0:
    print(num)
  #if it fails the above condition, it will
  else:
    continue
```

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
```

back

Ch1-ex08

```
for rows in range(1,6):

  for columns in range(1,rows+1): # Thi

    print(columns, end = " ") # We use

  print() # We call on the 'print()' st
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

back

Ch1-ex09 code

```
from random import seed
from random import randint

# We seed the later random number generator by giving it a random number from 0~10. If we do not give a seed to
seed(randint(0,10))

# 1. Initialize a list with 10 values with random numbers. Random numbers are determined by the integer the 'se
int_list = [randint(0,100) for i in range(10)]

# 2. We output the list using a 'for loop'. It has 10 values so we will loop for 10 times
index_to_print = 0 # We create a variable to be incremented by 1 to be used in the 'for loop'
print("This is the list as it is:", end = " ") # We change the 'end' value so that the numbers will be output i
for loop in range(10):
  print(int_list[index_to_print],end = ", ") # We change the 'end' value here as well so that the numbers are s
  index_to_print+=1

print("\n") # Create 2 newlines. Output ends with a newline by default. This will make it so that there is 1 em

# 3. To output our highest value, we use the 'max()' function. To output our lowest value, we use the 'min()' f
highest_value = max(int_list)
lowest_value = min(int_list)
print(f"The highest number in our 'int' list is '{highest_value}' and the lowest is '{lowest_value}'! \n")


# 4. To sort the elements in ascending order
sorted_list_asc = sorted(int_list)
print(f"This is the sorted list in asncending order: {sorted_list_asc}. \n")

# 5. To sort the elements in descending order
sorted_list_desc = sorted(int_list, reverse = True)
print(f"This is the sorted list in descending order: {sorted_list_desc}. \n")

# 6. For appending two elements, I will use a the 'randint()' function
for x in range(2):
  int_list.append(randint(0,100))

# 7. Then we print the list itself after appending
```

back

Ch1-ex09 output

```
This is the list as it is: 59, 78, 47, 34, 17, 23, 86, 0, 43, 64,

The highest number in our 'int' list is '86' and the lowest is '0'!

This is the sorted list in asncending order: [0, 17, 23, 34, 43, 47, 59, 64, 78, 86].

This is the sorted list in descending order: [86, 78, 64, 59, 47, 43, 34, 23, 17, 0].

This is the list after appending 2 elements: [59, 78, 47, 34, 17, 23, 86, 0, 43, 64, 59, 77].
```

back

Ch1-ex10

```
# We create a dictionary that stores data about the movie "Inceptio
film_details = {"Title": "Inception",
                "Director": "Christopher Nolan",
                "Year of release": 2010,
                "Rating" : 8.8,
                "Box Office" : "839 million"
               }
# We display the contents (keys and values) of the dictionary
for keys, values in film_details.items():
    if keys == "Title":
        print(f"The Movie's {keys} is {values}", end = ". ") # The
    else:
        print(f" The {keys} is {values}", end = ". ") # The printed
```

```
The Movie's Title is Inception.  The Director is Christopher Nolan.  The Year of
 release is 2010.  The Rating is 8.8.  The Box Office is 839 million.
```

back

Ch1-ex11 code

```
year = (2017,2003,2011,2005,1987,2009,2020,2018,2009)

# 2. We access the value at the given index (-3)
print(f"The 3rd-to-the-last number in the tuple is: {year[-3]}! \n") # This prints the 3rd-to-the-la

# 3. To reverse the tuple, we will use a slicing notation([::]). Tuples are immutable, so we can't d
reverse_year = year[::-1]

print(f"This is the original tuple: {year}.") # We print the original and the reversed copy
print(f"And this is the reversed tuple: {reverse_year}. \n")

# 4. We will use '.count()' to see how many elements match the integer '2009'
times_2009_appeared = year.count(2009)
print(f"The number of times '2009' has appeared in the tuple is: {times_2009_appeared}! \n")

# 5. We will use '.index()' to get the index value of the integer '2018'
index_2018 = year.index(2018)
print(f"The index value of '2018' in the tuple is: {index_2018}!")
print(f"This means that the number is in the {index_2018}th position! \n")

# 6. We will use len to find out how many elements are in the tuple
year_len = len(year)
print(f"The length of the 'year' tuple is: {year_len}!")
print(f"This means there are {year_len} elements in the tuple!")
```

back

Ch1-ex11 output

```
The 3rd-to-the-last number in the tuple is: 2020!

This is the original tuple: (2017, 2003, 2011, 2005, 1987, 2009, 2020, 2018, 200
9).
And this is the reversed tuple: (2009, 2018, 2020, 2009, 1987, 2005, 2011, 2003,
 2017).

The number of times '2009' has appeared in the tuple is: 2!

The index value of '2018' in the tuple is: 7!
This means that the number is in the 7th position!

The length of the 'year' tuple is: 9!
This means there are 9 elements in the tuple!
```

back

Ch1-ex12 code 1

```
from math import pi

# Initialize string variables. This will be the template of our sentences. For reuse purposes
start = "So, you've chosen to calculate the area of a {}! To calculate this, we need to {}."
input_param = "Enter a number to represent its"
result = "And the area of the {} with the parameters you entered is: {}!"

print(f"For this exercise, we will calculate the area of a square (1), circle (2), or triangle(3)! \n")

# Initialize an 'input' variable to determine which shape's area is to be calculated
entered_num = (input("Enter a number to pick which shape's area you'd like to calculate: \n"))
```

back

Ch1-ex12 code 2

```
print() # We want to create an empty space so our output looks cleaner
if entered_num == '1':

    # We initialize these variables because we will use the '.format()' method later
    shape = "square"
    method = "multiply its width by its height"

    print(f"{start.format(shape,method)} \n") # You can see from the 'start' variable

    width = int(input(f"{input_param} width: \n"))
    height = int(input(f"{input_param} height: \n"))

    area = width * height

    print() # We want to create an empty space so our output looks cleaner

    print(f"{result.format(shape,area)} \n")

elif entered_num == '2':
    shape = "circle"
    method = "multiply its radius by pi"

    print(f"{start.format(shape,method)} \n")

    radius = int(input(f"{input_param} radius: \n"))

    area = radius * pi

    print() # We want to create an empty space so our output looks cleaner

    print(f"{result.format(shape,area)} \n")

elif entered_num == '3':
    shape = "triangle"
    method = "multiply its base by its height"

    print(f"{start.format(shape,method)} \n")

    base = int(input(f"{input_param} base: \n"))
    height = int(input(f"{input_param} height: \n"))

    area = base * height

    print() # We want to create an empty space so our output looks cleaner

    print(f"{result.format(shape,area)} \n")
```

back

Ch1-ex12 output

```
For this exercise, we will calculate the area of a square (1), circle (2), or triangle(3)!

Enter a number to pick which shape's area you'd like to calculate:
3

So, you've chosen to calculate the area of a triangle! To calculate this, we need to multiply its base by its height.

Enter a number to represent its base:
2
Enter a number to represent its height:
4

And the area of the triangle with the parameters you entered is: 8!
```

back

Ch1-ex13 code

```
from random import seed
from random import randint

def list_product(my_list):

    # We will call the 'seed()' function to seed our 'randint()' functions. The
    seed(randint(0,10))

    # Initialize a counter for the total product. Putting this variable outside
    product = 1

    print(f"These are the numbers in that list: {rand_list}! \n")

    for i in my_list:

        # We can use a variable's previous value and assign it to the same varia
        product = product * i

    return product

rand_list = [randint(1,10) for i in range(3)]

print(f"And the product of all the numbers is... {list_product(rand_list)}!")
```

back

Ch1-ex13 output

```
These are the numbers in that list: [3, 7, 6]!

And the product of all the numbers is... 126!
```

back

Ch1-bonus-ex01

```
# Create a loop that ranges from numbers 1~10. Th
for x in range(1,11):
  print(f"This is the table of {x}!")

  # We create a nested loop ranging from numbers
  for y in range(1,11):
    print(f"{x} x {y} = {x*y}!")

  # We put a 'print()' statement to create an emp
  print()
```

```
This is the table of 1!
1 x 1 = 1!
1 x 2 = 2!
1 x 3 = 3!
1 x 4 = 4!
1 x 5 = 5!
1 x 6 = 6!
1 x 7 = 7!
1 x 8 = 8!
1 x 9 = 9!
1 x 10 = 10!

This is the table of 2!
2 x 1 = 2!
2 x 2 = 4!
2 x 3 = 6!
2 x 4 = 8!
2 x 5 = 10!
2 x 6 = 12!
2 x 7 = 14!
2 x 8 = 16!
2 x 9 = 18!
2 x 10 = 20!

This is the table of 3!
3 x 1 = 3!
3 x 2 = 6!
3 x 3 = 9!
3 x 4 = 12!
3 x 5 = 15!
3 x 6 = 18!
3 x 7 = 21!
3 x 8 = 24!
3 x 9 = 27!
3 x 10 = 30!

This is the table of 4!
4 x 1 = 4!
4 x 2 = 8!
4 x 3 = 12!
4 x 4 = 16!
4 x 5 = 20!
4 x 6 = 24!
4 x 7 = 28!
4 x 8 = 32!
4 x 9 = 36!
4 x 10 = 40!
```

back

Ch1-bonus-ex02 code

```
locations =['dubai','paris', 'switzerland', 'London', 'amsterdam', 'New York']

# 1. We print the list as is and its length
locations_len = len(locations)
print(f"The list contains: {locations}. And the length of this list is {locations_len}! \n")

# 2. We sort the list
locations_sorted = sorted(locations)
print(f"When we use the 'sorted()' function on our list, we get will this list: {locations_sorted}! \n")

# 3. We will show the original list. The 'sorted()' function does not modify the list itself, rather it returns a different iist but sorted based on the
print(f"The original list remains intact. As a matter of fact, here it is!: {locations}! \n")

# 4. We will sort the list in reverse order
locations_sorted_reverse = sorted(locations, reverse = True)
print(f"The list in reverse alphabetical order is.. {locations_sorted_reverse}! \n")

# 5. Print the original list. The og list is still unaffected
print(f"The list is still intact! Here it is!: {locations}! \n")

# 6. Use the '.reverse()' method. This will arrange the list in reverse order (not ascending or descending), and modify the og list and it returns 'None
locations.reverse()

# 7. Print the og list
print(f"The list has now been permanently reversed! This is now the original list: {locations}! \n")

# 8. Use '.sort()' method to sort the list in ASCII-betical order
locations.sort()

# 9. Print the list
print(f"The old order of the list is gone, it is now sorted not in reverse, but in ascending order (In terms of its ASCII value)!...: {locations}! \n")

# 10. Use '.sort()' but make it so the it arranges it from reverse ASCII-betical order
locations.sort(reverse=True)

# 11. Print the list
print(f"And for my last trick, I will sort the list in reverse ASCII-betical order (Descending ASCII values)!... This is it!: {locations}! \n")
```

back

Ch1-bonus-ex02 output

```
The list contains: ['dubai', 'paris', 'switzerland', 'London', 'amsterdam', 'New
 York']. And the length of this list is 6!

When we use the 'sorted()' function on our list, we get will this list: ['London
', 'New York', 'amsterdam', 'dubai', 'paris', 'switzerland']!

The original list remains intact. As a matter of fact, here it is!: ['dubai', 'p
aris', 'switzerland', 'London', 'amsterdam', 'New York']!

The list in reverse alphabetical order is.. ['switzerland', 'paris', 'dubai', 'a
msterdam', 'New York', 'London']!

The list is still intact! Here it is!: ['dubai', 'paris', 'switzerland', 'London
', 'amsterdam', 'New York']!

The list has now been permanently reversed! This is now the original list: ['New
 York', 'amsterdam', 'London', 'switzerland', 'paris', 'dubai']!

The old order of the list is gone, it is now sorted not in reverse, but in ascen
ding order (In terms of its ASCII value)!...: ['London', 'New York', 'amsterdam'
, 'dubai', 'paris', 'switzerland']!

And for my last trick, I will sort the list in reverse ASCII-betical order (Desc
ending ASCII values)!... This is it!: ['switzerland', 'paris', 'dubai', 'amsterd
am', 'New York', 'London']!
```

back

Ch1-bonus-ex03 code 1

```
def add(x,y):
  return x + y

def subtract(x,y):
  return x - y

def multiply(x,y):
  return x * y

def divide(x,y):
  return x / y

def modulo(x,y):
  return x % y

# We will also make it a function to enter our two numbers
def operation():
  num_1 = int(input("Enter a number to represent our first number: \n"))
  num_2 = int(input("Enter a number to represent our second number: \n"))
  return num_1, num_2

# We will also make a string variable with placeholders for code reuse
result_txt = "The result of {} and {} performing {} is: {}!"

operations_txt = "Addition (1), Subtraction (2), Multiplication (3), Division (4), Modulus (5)"
start_txt = [f"For this exercise, we are going to create a 'calculator' program! Simply enter what number you'd like to operate: {operations_txt}",
             f"Enter what number you'd like to operate: {operations_txt}"]

# Initialize a counter to change texts when we loop for another time
iterations = 0

exit_main_loop = False

# We will put our code inside a 'while' loop
while exit_main_loop != True:

 exit_sec_loop = False

 if iterations == 0:
  enter_num = input(f"{start_txt[iterations]} \n")
 else:
  enter_num = input(f"{start_txt[iterations]} \n")

 if enter_num == '1':
  # Make a string variable to pass later on
  operation_str = "addition"
```

back

Ch1-bonus-ex03 code 2

```
if enter_num == '1':
 # Make a string variable to pass later on
 operation_str = "addition"

 # We assign the 2 returned values to these variables
 first_num, sec_num = operation()

 result = add(first_num,sec_num) # Call the 'add()' function as we pass these 2 v

 # We pass 4 variables to replace our 4 placeholders using '.format()' method
 print(f"{result_txt.format(first_num,sec_num,operation_str,result)} \n")

elif enter_num == '2':
 operation_str = "subtraction"

 first_num, sec_num = operation()

 result = subtract(first_num,sec_num)

 print(f"{result_txt.format(first_num,sec_num,operation_str,result)} \n")

elif enter_num == '3':
 operation_str = "multiplication"

 first_num, sec_num = operation()

 result = multiply(first_num,sec_num)

 print(f"{result_txt.format(first_num,sec_num,operation_str,result)} \n")

elif enter_num == '4':
 operation_str = "divison"

 first_num, sec_num = operation()

 result = divide(first_num,sec_num)

 print(f"{result_txt.format(first_num,sec_num,operation_str,result)} \n")

elif enter_num == '5':
  operation_str = "modulo division"

  first_num, sec_num = operation()

  result = modulo(first_num,sec_num)

  print(f"{result_txt.format(first_num,sec_num,operation_str,result)} \n")
```

back

Ch1-bonus-ex03 code 3

```
    print(f"{result_txt.format(first_num,sec_num,operation_str,result)} \n")

 while True:
  try_again = input("Would you like to perform another calculation? (y/n) \n")

  if try_again == 'y':
   iterations = 1
   break # Leave the second loop, which will make up go back to the top of the

  elif try_again == 'n':
   exit_main_loop = True # This will fail the condition when the program inevit
   break # Exit this second loop

  else:
   continue # You will be stuck in this loop until you enter either 'y' or 'n'

   print() # We will create an empty space after every iteration

print("You've exited the loop!")
```

back

Ch1-bonus-ex03 output

```
For this exercise, we are going to create a 'calculator' program! Simply enter w
hat number you'd like to operate: Addition (1), Subtraction (2), Multiplication
(3), Division (4), Modulus (5)
3
Enter a number to represent our first number:
25
Enter a number to represent our second number:
86
The result of 25 and 86 performing multiplication is: 2150!

Would you like to perform another calculation? (y/n)
a
Would you like to perform another calculation? (y/n)
a
Would you like to perform another calculation? (y/n)
y
Enter what number you'd like to operate: Addition (1), Subtraction (2), Multipli
cation (3), Division (4), Modulus (5)
7
Would you like to perform another calculation? (y/n)
y
Enter what number you'd like to operate: Addition (1), Subtraction (2), Multipli
cation (3), Division (4), Modulus (5)
1
Enter a number to represent our first number:
2
Enter a number to represent our second number:
7
The result of 2 and 7 performing addition is: 9!

Would you like to perform another calculation? (y/n)
n
You've exited the loop!
```

back

Ch1-further-ex01 code

```
print("In this exercise, we will turn the amount of days you give into seconds! \n")

# Initialize boolean to be used later
exit_loop = False

while exit_loop == False:

    # Initialize an input variable
    days = int(input("Enter the amount of days: \n"))

    days_to_seconds = days * 24 * 60 * 60

    print(f"The amount of seconds in the amount of days you entered is...{days_to_seconds}! \n")

    while True:

        try_again = input("Would you like to continue? (y/n) \n") # We put this code inside the second loop b

        if try_again == 'y':
            break

        elif try_again == 'n':
            exit_loop = True
            break

        else:
            print("Invalid character")
```

back

Ch1-further-ex01 output

```
In this exercise, we will turn the amount of days you give into seconds!

Enter the amount of days:
20
The amount of seconds in the amount of days you entered is...1728000!

Would you like to continue? (y/n)
y
Enter the amount of days:
1
The amount of seconds in the amount of days you entered is...86400!

Would you like to continue? (y/n)
n
```

back

Ch1-further-ex02 code

```
# For this exercise, we will make a function
def sum_of_digits(number):
    current_sum = 0
    while number > 0:

        current_sum += number % 10 # We will collect the number in the ones-place from the tens-pl

        number = number//10 # We remove the current ones-place because it has already been added t


    return current_sum

num = int(input("Enter a number, and this program will find the sum of its digits! \n"))

print(f"The sum of the digits in {num} is... {sum_of_digits(num)}!")
```

back

Ch1-further-ex02 output

```
Enter a number, and this program will find the sum of its digits!
1234
The sum of the digits in 1234 is... 10!
```

back

Ch1-further-ex03

```
times=4
for x in range(8):
    if x < 5:
        spaces = " " * times
        star = "*" * (x * 2 + 1)
        print(spaces+star)
        times -=1
    else:
        spaces = " " * 3
        star  = "*" * 3
        print(spaces+star)
```

```
    *
   ***
  *****
 *******
*********
   ***
   ***
   ***
```

back

Ch1-further-ex04 code 1

```
# We will initialize a boolean variable as we will need to control the flow of the loops later on. We will use 2 loops, a main one, and a nested
outer_loop = True
inner_loop = True

# Initialize a string list of names be checked which input is valid
staff = ["Arshiya", "Usman", "Iftikhar", "Usman","Rafia", "Mary", "Anmol","Zainab","Iftikhar", "Arshiya","Rafia","Jake"]

# We make a function here because 'why not?'
def response_take(yn):
  global outer_loop # The names after the keyword 'global' are referring to the global variables. Meaning the variable is defined outside the function
  global inner_loop
  function_loop = True # This is a local variable. Meaning it is only defined inside the function. The only reason why we have this is for the 'else'

  while function_loop == True:
    if yn == 'y':
      inner_loop = False # This means for the next iteration, the inner loop's condition would be false
      function_loop = False # This means that we will not run the next iteration of this function, causing us to exit the function since there is none

    elif yn == 'n':
      # We fail all the conditions of all the loops so that we exit all the loop code-blocks in the next iteration (there won't be any)
      inner_loop = False
      outer_loop = False
      function_loop = False

    else:
      yn=input("Invalid response! Would you like to try again? (y/n) \n") # I used a local variable here because we want the function to keep updating


print(f"For this exercise, we will count the amount of times a specific name appears in the list!\n")

while outer_loop == True:

  # Initialize this boolean var (inner_loop) inside the main 'while loop' so that its value resets per iteration
  inner_loop = True

  name = input("Enter whose name you'd like to count: 'Arshiya', 'Usman', 'Iftikhar', 'Rafia', 'Mary , 'Anmol', 'Zainab', 'Jake' \n")

  # We turn the input name into uppercase so that if it's a lowercase input, it is still valid
  input_name = name.title()

  # Check if the uppercased input matches any items in the list
  if input_name in staff:

    # Store the count inside a variable for efficiency (we don't have to type 'staff.count(input_name)' repeatedly)
    name_count = staff.count(input_name)
```

back

Ch1-further-ex04 code 2

```
    print(f"The amount of times the name '{input_name}' appears on the list is... {name_count}!!! \n")
    print(f"The list: {staff}\n")
    response = input("That's all for this program! Would you like to try again? (y/n) \n")

    # We call on the function, passing an input var as the argument
    response_take(response)

  else:
    while inner_loop == True:
      response=input("That name is not on the list, would you like to try again? (y/n) \n")
      response_take(response)

print("Thanks for playing!")
```

back

Ch1-further-ex04 output

```
For this exercise, we will count the amount of times a specific name appears in the list!

Enter whose name you'd like to count: 'Arshiya', 'Usman', 'Iftikhar', 'Rafia', 'Mary , 'Anmol', 'Zainab', 'Jake'
usman
The amount of times the name 'Usman' appears on the list is... 2!!!

The list: ['Arshiya', 'Usman', 'Iftikhar', 'Usman', 'Rafia', 'Mary', 'Anmol', 'Zainab', 'Iftikhar', 'Arshiya', 'Rafia', 'Jake']

That's all for this program! Would you like to try again? (y/n)
a
Invalid response! Would you like to try again? (y/n)
y
Enter whose name you'd like to count: 'Arshiya', 'Usman', 'Iftikhar', 'Rafia', 'Mary , 'Anmol', 'Zainab', 'Jake'
b
That name is not on the list, would you like to try again? (y/n)
f
Invalid response! Would you like to try again? (y/n)
y
Enter whose name you'd like to count: 'Arshiya', 'Usman', 'Iftikhar', 'Rafia', 'Mary , 'Anmol', 'Zainab', 'Jake'
mary
The amount of times the name 'Mary' appears on the list is... 1!!!

The list: ['Arshiya', 'Usman', 'Iftikhar', 'Usman', 'Rafia', 'Mary', 'Anmol', 'Zainab', 'Iftikhar', 'Arshiya', 'Rafia', 'Jake']

That's all for this program! Would you like to try again? (y/n)
n
Thanks for playing!
```

back

Ch1-further-ex05

```
# We initialize the "marks" variable
marks = [("CodeLab I",67),("web Development", 75),("CodeLabII",74),("Smartphone Apps",68),("Games Development",70),("Responsive web",65)]
print(f"This is the 'marks' variable by default!: {marks} \n")

# We will use the 'sorted()' function with a custom key to sort by their tuple's second elements
marks_descending = sorted(marks,key = lambda item: item[1], reverse = True)
print(f"This is 'marks' arranged in descending by their second elements!: {marks_descending} \n")

# We will sort by ascending, but using the same custom key
marks_ascending = sorted(marks, key = lambda item: item[1])
print(f"This is 'marks' arranged in ascending order by their second elements!: {marks_ascending} \n")
```

```
This is the 'marks' variable by default!: [('CodeLab I', 67), ('web Development'
, 75), ('CodeLabII', 74), ('Smartphone Apps', 68), ('Games Development', 70), ('
Responsive web', 65)]

This is 'marks' arranged in descending by their second elements!: [('web Develop
ment', 75), ('CodeLabII', 74), ('Games Development', 70), ('Smartphone Apps', 68
), ('CodeLab I', 67), ('Responsive web', 65)]

This is 'marks' arranged in ascending order by their second elements!: [('Respon
sive web', 65), ('CodeLab I', 67), ('Smartphone Apps', 68), ('Games Development'
, 70), ('CodeLabII', 74), ('web Development', 75)]
```

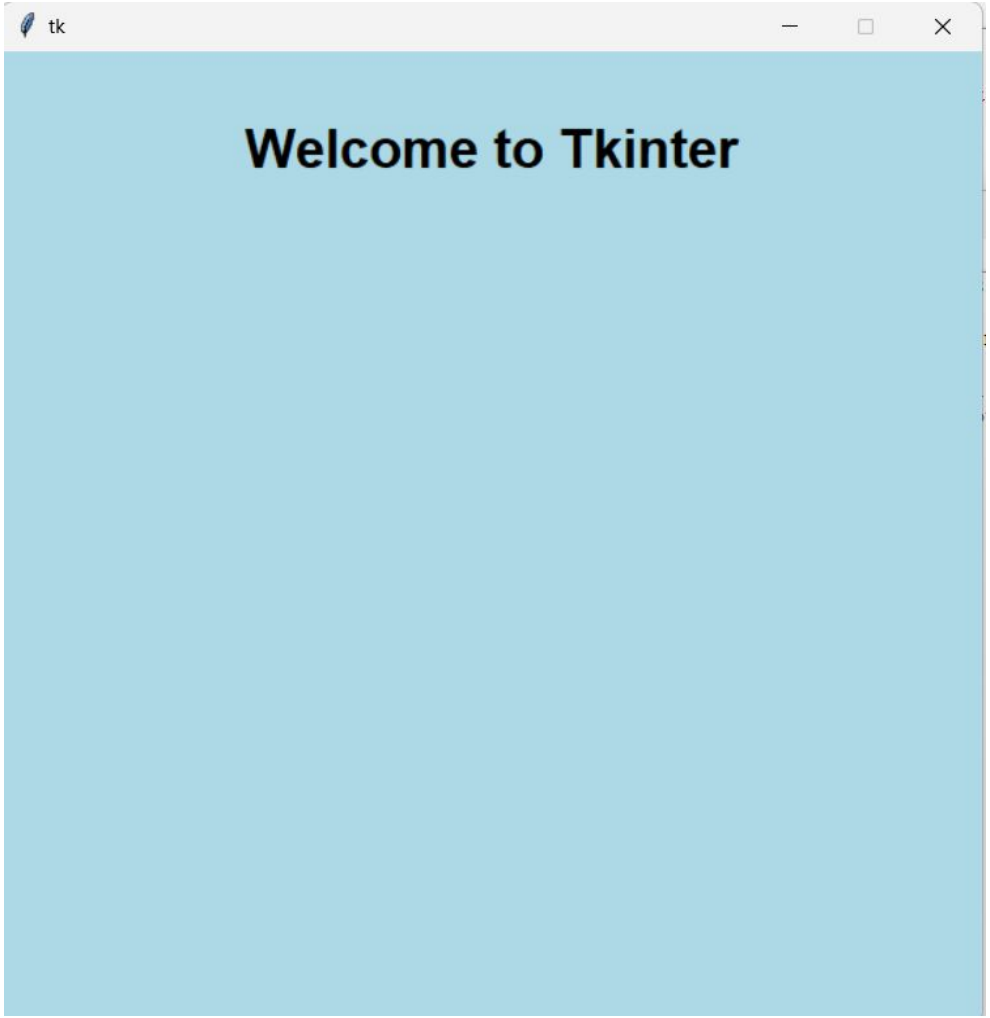
back

Ch2-ex01 code

```
# Import module
import tkinter as tk

# Create an output window
main = tk.Tk()

# Set a base window size
main.geometry('600x600')

#(2)
main.resizable(0,0)

#(1)
text = tk.Label(main,text ='Welcome to Tkinter', font=('Roboto',25, 'bold'),bg = 'lightblue')

# Place the label on the app
text.place(relx=.5,rely=.1, anchor = 'center')

#(3)
main['bg'] = 'lightblue'

# Start the app
main.mainloop()
```

back

Ch2-ex01 output



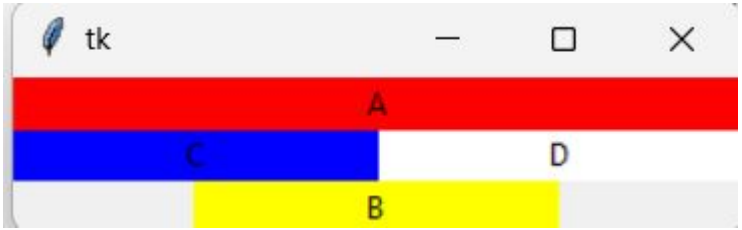
back

Ch2-ex02-1

```
# Import tkinter module
import tkinter


# Create an output window
root= tkinter.Tk()

# Set a base resize window
root.geometry('600x600')

# Create the labels
letter_A=tkinter.Label(root, text='A', bg='red')
letter_B=tkinter.Label(root, text='B', bg='yellow', width = 20)
letter_C=tkinter.Label(root, text='C', bg='blue', width = 20)
letter_D=tkinter.Label(root, text='D', bg='white', width = 20)

# Output the labels
# 'anchor' default: 'center' after applying 'side'
letter_A.pack(side=tkinter.TOP, fill = tkinter.X, expand = tkinter.YES)

letter_B.pack(side=tkinter.BOTTOM)

letter_C.pack(side = tkinter.LEFT,expand=tkinter.YES)

letter_D.pack(side=tkinter.RIGHT)
# Start the app
root.mainloop()
```

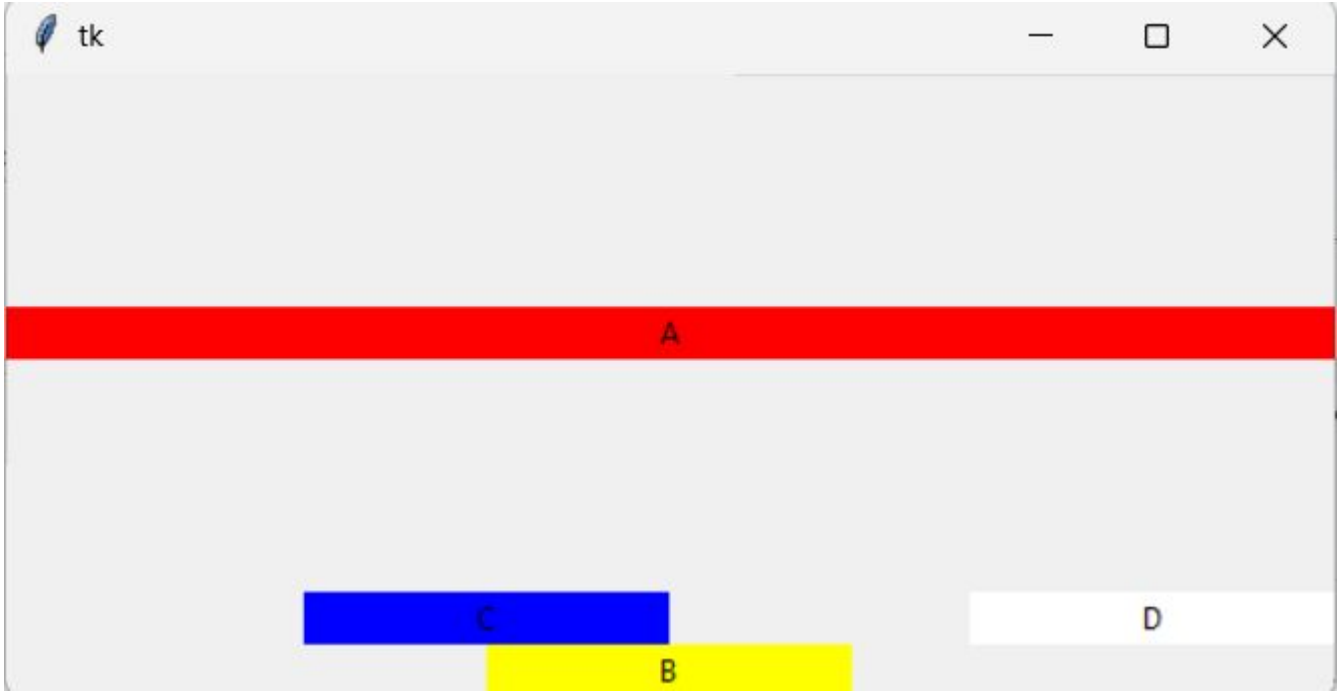
back

Ch2-ex02-1 output 1



back

Ch2-ex02-1 output 2



back

Ch2-ex02-2 code 1

```
import tkinter

# Create root window
main = tkinter.Tk()

# Create window size
main.geometry('600x600')

# Change default title
main.title('GUI pack sample')

# customizable options
l_preset={'bg':'white','relief':'groove'}
r_preset={'bg':'white','relief':'sunken'}

left_frame = tkinter.Frame(main,bd=5,**l_preset)
right_frame=tkinter.Frame(main,bd=5,**r_preset)

# Output the frame. Must have set sizes to be visible
left_frame.place(relwidth=.5,relheight=1)
right_frame.place(relwidth=.5,relheight=1,relx=.5)


# list to store the 4 labels
labels=[]

# loop to create the 4 labels
for i in range(4):
    # first 2 iterations go to the left frame
    if i<2:
        l=tkinter.Label(left_frame)
        labels.append(l)
    else:
        l=tkinter.Label(right_frame)
        labels.append(l)

# number for increment
index=0

# list for iteration
letters=['A','B','C','D']

# customize the labels stored in the 'labels' list
for i in labels:

    # all labels will have their text from here
```

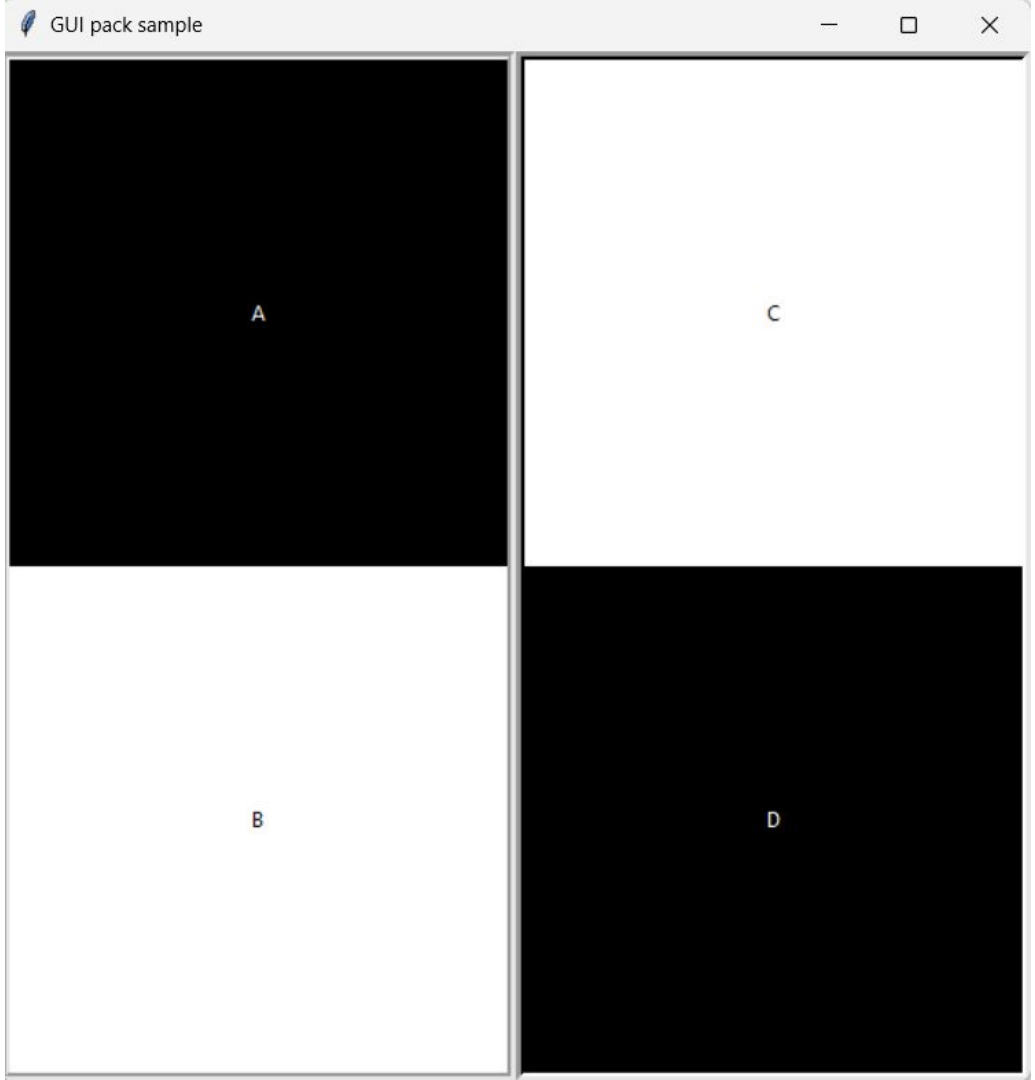
back

Ch2-ex02-2 code 2

```
    # all labels will have their text from here
    i.configure(text=f'{letters[index]}')
    index+=1

    # set colors of the labels and output them in their correct positions
    if labels.index(i) == 0 or labels.index(i)==3:
        i.configure(bg='black',fg='white')
        i.pack(expand=tkinter.YES,fill=tkinter.BOTH,anchor='n')
    else:
        i.configure(bg='white')
        i.pack(expand=tkinter.YES,fill=tkinter.BOTH,anchor='s')


# Start the app. All codes must be before this line
main.mainloop()
```

back

Ch2-ex02-2 output



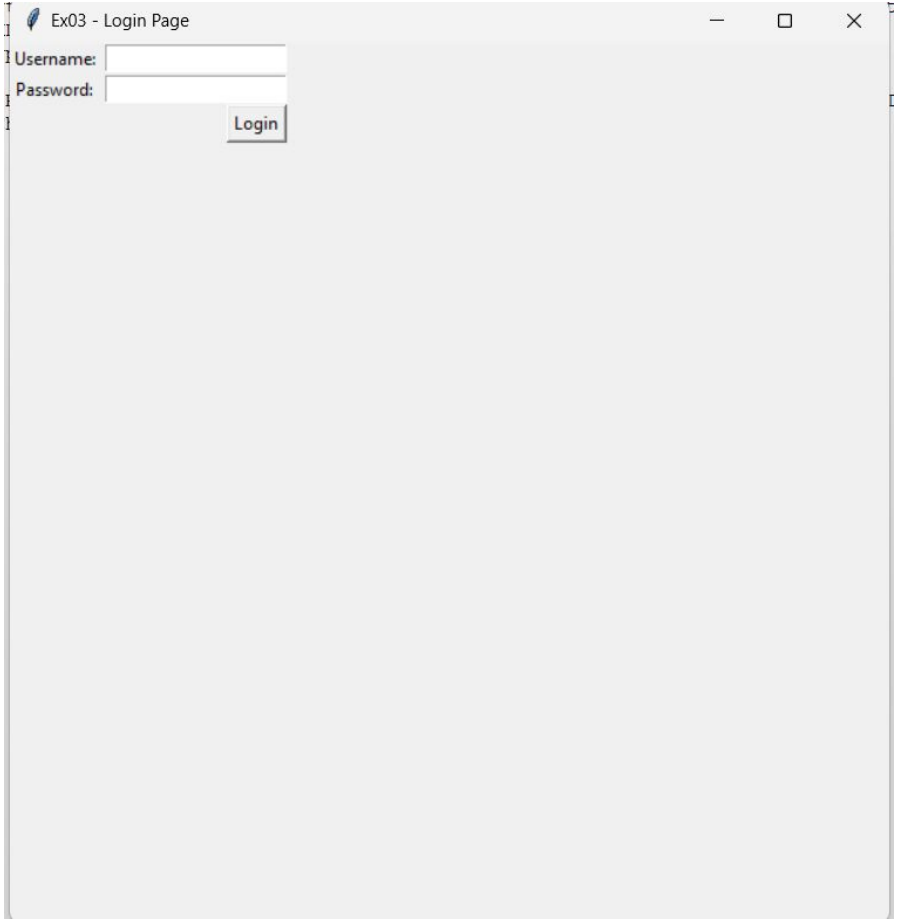
back

Ch2-ex03 code

```
import tkinter as tk

# create an output window
main = tk.Tk()

# set title
main.title('Ex03 - Login Page')

# set base window size
main.geometry('600x600')

# row 1
tk.Label(main,text='Username: ').grid()

# specify row else it will be placed in {previous widget's row} + 1. doesn't apply to columns ???
tk.Entry(main).grid(column=1,row=0)

# row 2
tk.Label(main,text='Password: ').grid(row=1)

tk.Entry(main).grid(row=1,column=1)

# row 3
# grid has sticky intead of anchor
tk.Button(main,text='Login').grid(sticky='e',row=3,column=1)
# start the app
main.mainloop()
```

back

Ch2-ex03 output



back

Ch2-ex04 code 1

```
import tkinter
from PIL import ImageTk
from tkinter import ttk, messagebox

# We do this using grid because 'why not?'

# create output window
root=tkinter.Tk()

# set a base window size
root.geometry('413x620')

# make it unresizable
root.resizable(0,0)

# set a title
root.title('Ex04 Registration Page with grid geometry')

# set default bg
root.configure(bg='white')

# notes:
#   total rows: 17
#   total columns: 3


#-------Functions--------
# In rows 1~16
def font(size=12,**kwargs):
    if 'b' in kwargs:
        return ('Tahoma',size,'bold')
    else:
        return ('Tahoma',size)

# In row 15
# A function can have not-yet-defined variable perform an action. The error comes when calling
def slider_changed(event):
    l11.configure(text=get_current_value())

def get_current_value():
    return current_value.get()

# In row 16
def submitcheck():
    entry_len=[]
    for i in entry_dict:
        entry_len.append(len(entry_dict[i].get()))
```

back

Ch2-ex04 code 2

```
        entry_len.append(len(entry_dict[i].get()))

    if 0 not in entry_len:
        messagebox.showinfo('Success','You have submitted your form to nowhere!')

    else:
        messagebox.showerror('Error','You must fill all entry fields')

    global easter_egg
    if easter_egg==4:
        easter_egg=0

    easter_egg_l=["I'm a buff baby, but I dance like a man","She a nice lady and she shakin' the
    print(easter_egg_l[easter_egg])
    easter_egg+=1

# Deletes anything entered in the Entry widget
def on_clear():
    global e1,e2,e3,e4,e5
    all_entries=[e1,e2,e3,e4,e5]

    for i in all_entries:
        i.delete(0,'end')

#------Variables---------
label_n_checkb_preset= {'font':font,'bg':'#f5f5f6'}
entry_preset={'font':font,'bg':'#adaeb7'}
radiob_preset={'font':font(11),'bg':'#f5f5f6'}
button_preset={'font':font,'width':8,'bg':'#22263d','fg':'white'}

# Create a dict for all the Entry widgets. To be stored by the widgets' 'textvariable' argument
entry_dict = {}
for i in range(1,6):
    entry_dict[f'entry{i}']=tkinter.StringVar()

# Variable to store an integer. In row 15
current_value = tkinter.IntVar()

easter_egg=0
#----------------------
# row 0
# create image
banner_img=ImageTk.PhotoImage(file='Images/BSULOGO.png')
# output image
tkinter.Label(root,image=banner_img).grid(columnspan=3)

# row 1
```

back

Ch2-ex04 code 3

```
# row 1
# Frame
frame=tkinter.Frame(root,bg='#f5f5f6')
frame.grid(row=1,rowspan=16,columnspan=3,padx=10)

l1=tkinter.Label(frame,text='Student Management System', font=font(18,b='yes'))
l1.grid(row=1,columnspan=3)

# row 2
l2=tkinter.Label(frame,text='New Student Registration',font=font(15,b='yes'))
l2.grid(row=2,columnspan=3)

# row 3
l3=tkinter.Label(frame,text='Student Name',**label_n_checkb_preset)
l3.grid(row=3,sticky='w')

e1=tkinter.Entry(frame,**entry_preset,textvariable=entry_dict['entry1'])
e1.grid(row=3,column=1,sticky='w',columnspan=2)

# row 4
l4=tkinter.Label(frame,text='Mobile Number',**label_n_checkb_preset)
l4.grid(row=4,sticky='w')

e2=tkinter.Entry(frame,**entry_preset,textvariable=entry_dict['entry2'])
e2.grid(row=4,column=1,sticky='w',columnspan=2)

# row 5
l5=tkinter.Label(frame,text='Email Id',**label_n_checkb_preset)
l5.grid(row=5,sticky='w')

e3=tkinter.Entry(frame,**entry_preset,textvariable=entry_dict['entry3'])
e3.grid(row=5,column=1,sticky='w',columnspan=2)

# row 6
l6=tkinter.Label(frame,text='Home Address',**label_n_checkb_preset)
l6.grid(row=6,sticky='w')

e4=tkinter.Entry(frame,**entry_preset,textvariable=entry_dict['entry4'])
e4.grid(row=6,column=1,sticky='w',columnspan=2)

# row 7
l7=tkinter.Label(frame,text='Gender',**label_n_checkb_preset)
l7.grid(row=7,sticky='w')

e5=tkinter.Entry(frame,**entry_preset,textvariable=entry_dict['entry5'])
e5.grid(row=7,column=1,sticky='w',columnspan=2)
```

back

Ch2-ex04 code 4

```
e5=tkinter.Entry(frame,**entry_preset,textvariable=entry_dict['entry5'])
e5.grid(row=7,column=1,sticky='w',columnspan=2)

# row 8
l8=tkinter.Label(frame,text='Course Enrolled',**label_n_checkb_preset)
l8.grid(row=8,sticky='w')

r1=tkinter.Radiobutton(frame,text='BSc CC',**radiob_preset,value=0)
r1.grid(row=8,sticky='w',column=1)

# row 9
r2=tkinter.Radiobutton(frame,text='BSc CY',**radiob_preset,value=1)
r2.grid(row=9,sticky='w',column=1)

# row 10
r3=tkinter.Radiobutton(frame,text='BSc PSY',**radiob_preset,value=2)
r3.grid(row=10,sticky='w',column=1)

# row 11
r4=tkinter.Radiobutton(frame,text='BA & BM',**radiob_preset,value=3)
r4.grid(row=11,sticky='w',column=1)

# row 12
l9=tkinter.Label(frame,text='Languages Known',**label_n_checkb_preset)
l9.grid(row=12,sticky='w')

c1=tkinter.Checkbutton(frame,text='English',**label_n_checkb_preset)
c1.grid(row=12,column=1,sticky='w')

c2=tkinter.Checkbutton(frame,text='Tagalog',**label_n_checkb_preset)
c2.grid(row=12,column=2)

# row 13
c3=tkinter.Checkbutton(frame,text='Hindu/Urdu',**label_n_checkb_preset)
c3.grid(row=13,column=1,sticky='w')

# row 14
# sticky is centered by default (???). 'we' does the same thing
l10=tkinter.Label(frame,text='Rate your English communication skills',**label_n_checkb_preset)
l10.grid(row=14,columnspan=3)

# row 15
# tkinter.scale orientation is vertical by default
slider = ttk.Scale(frame,from_=0,to=10,variable=current_value,command=slider_changed)
slider.grid(row=15,columnspan=2)

# In the case of creating objects, always separate the geometry management from the object, ESPE
```

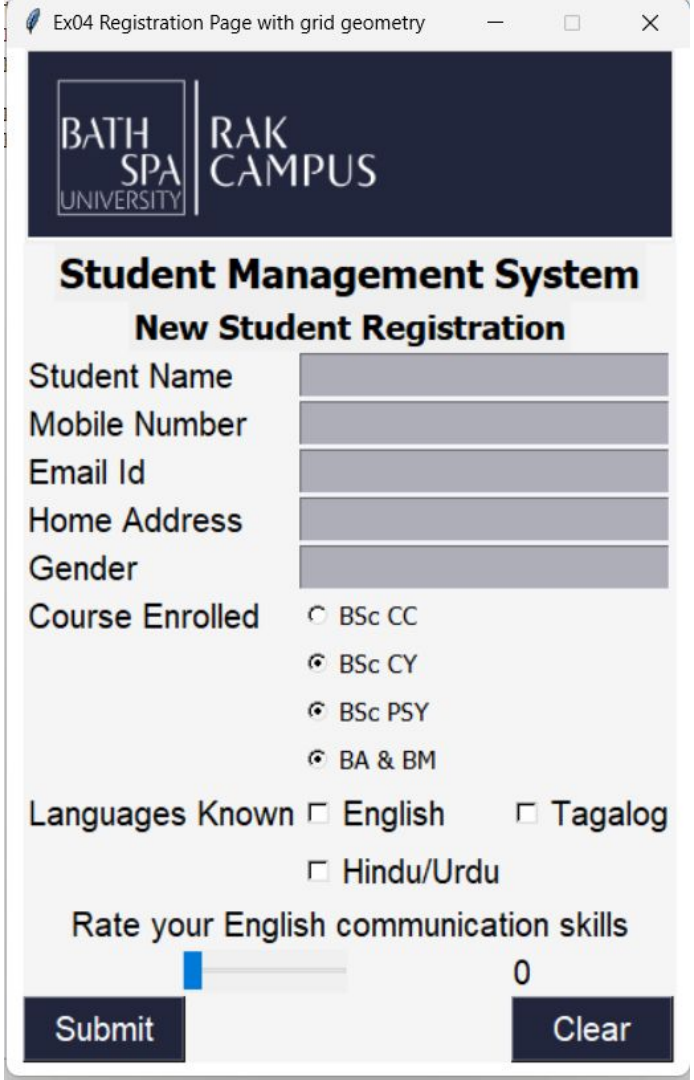
back

Ch2-ex04 code 5

```
# In the case of creating objects, always separate the geometry management from the object, ESPECIALLY when there is a function
# display the value of the Scale
l11 = tkinter.Label(frame,text=get_current_value(),**label_n_checkb_preset)
l11.grid(row=15, column=2,sticky='w')

# row 16
tkinter.Button(frame,text='Submit',**button_preset,command=submitcheck).grid(row=16,sticky='w')
tkinter.Button(frame,text='Clear',**button_preset,command=on_clear).grid(row=16,column=2)

# start the app
root.mainloop()
```

back

Ch2-ex04 output



back

Ch2-ex05 code 1

```
import tkinter as tk

# root window
root=tk.Tk()
root.geometry('370x600')
root.title('Calculator')
root.resizable(0,0)

def onclick(character):
    current_text= screen.cget('text')
    screen.configure(text=current_text + str(character))

    print(screen.cget('text'))
    print(type(screen.cget('text')))

def calculate(expression):
    output= str(eval(expression))
    screen.configure(text=output)

    print(screen.cget('text'))
    print(type(screen.cget('text')))

def clear():
    screen.configure(text='')

# app (grid starts at 0)

# screen of the calc
screen=tk.Label(root,bg='white',height=5,font=('Tahoma',12,'bold'))
screen.grid(row=0,columnspan=4,sticky='we')

# var here for iteration purposes
n=0

# 2 different lists to be iterated separately
symbols=['+','-','*','/']
symbols_2=['9','%','/']

# these will be the 3x3 from numbers 0 to 8
for x in range(1,4):
    for y in range(3):
        # we need to store in a lambda because it is an iteration, all the values will be on the latest itera
        tk.Button(root,text=str(n),width=12,height=6,command= lambda a=n: onclick(a)).grid(row=x,column=y)
        n+=1
# var here to iterate through indexes
i=0
```

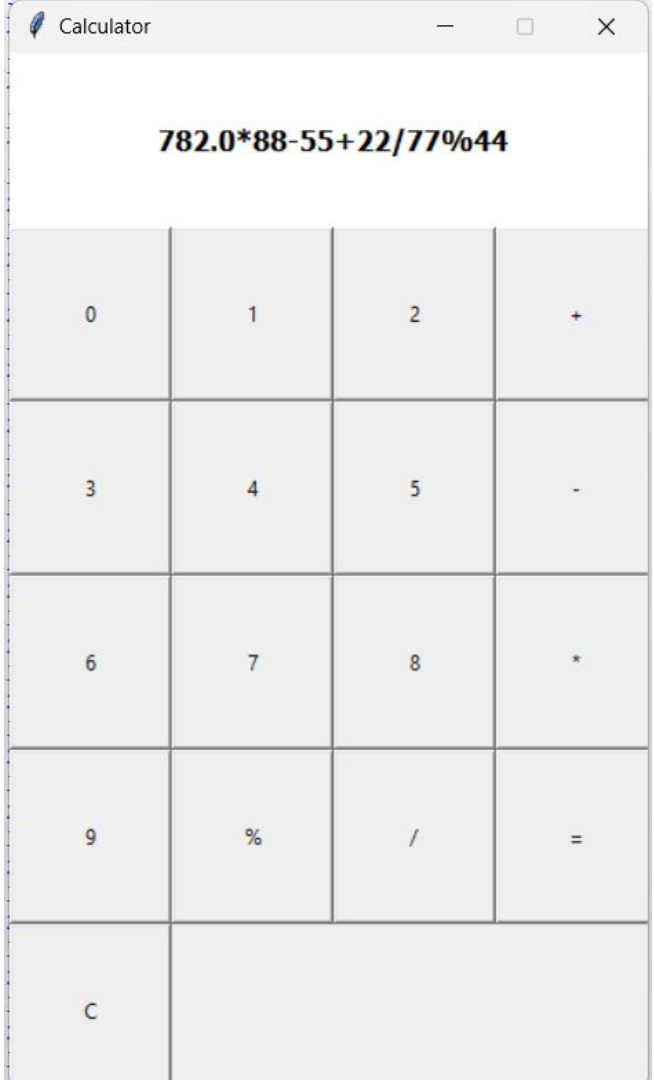
back

Ch2-ex05 code 2

```
# this will be the loop for the operator symbols
for x in range(1,4):
    tk.Button(root,text=symbols[i],width=12,height=6,command= lambda a=symbols[i]: onclick(a)).grid(row=x,column=3)
    i+=1 # increment by 1, so it will go through a different elements

# reset to zero for the code below
i=0

# loop for the remaining symbols
for y in range(3):
    tk.Button(root,text=symbols_2[i],width=12,height=6,command= lambda a=symbols_2[i]: onclick(a)).grid(row=4,column=y)
    i+=1

# these two are not included in the loop because of their command attribute
tk.Button(root,text='C',width=12,height=6,command= clear).grid(row=5)
tk.Button(root,text='=',width=12,height=6,command= lambda: calculate(screen.cget('text'))).grid(row=4,column=3)

# start app
root.mainloop()
```

back

Ch2-ex05 output



back

Ch2-bonus-ex01 code 1

```
import tkinter as tk
from PIL import ImageTk, Image
import re

def switch():
    global first_frames,second_frames
    for i in second_frames:
        i.tkraise()
    first_frames,second_frames=second_frames,first_frames

def ctoF(*args):
    if re.match(r'^\d+$',l_input.get()):
        c = float(l_input.get())
        f = c * 9 / 5 + 32
        fahrenheit.set(f'{f:.2f}')
    else:
        pass

def ftoC(*args):
    if re.match(r'^\d+$',l_input.get()):
        f=float(l_input.get())
        c= (f-32)*5/9
        celsius2.set(f'{c:.2f}')
    else:
        pass
# root window
root = tk.Tk()

root.geometry('400x400')
root.resizable(0, 0)
root.title('Temperature Converter')

# app (column, left to right)
base_img = Image.open('images/temperature-thermometer-icon-free-vector.jpg')
resize_img = base_img.resize((400, 400))
final_img = ImageTk.PhotoImage(resize_img)

background = tk.Label(root, image=final_img)
background.place(x=0)

def font(**kwargs):
    size=kwargs.get('size',20)
    font=('Tahoma',size)

    return font
```

back

Ch2-bonus-ex01 code 2

```
    return font


# frame 1
f1=tk.Frame(root,bg='white')
f1.place(relwidth=.35,relheight=1)

l1 = tk.Label(f1, text='Celsius', font=font(), background='white')
l1.place(relx=.5, rely=.1,anchor='c')

l_input = tk.StringVar()
l_input.trace_add('write', ctoF)
e1 = tk.Entry(f1, font=font(), bg='lightgray', textvariable=l_input)
e1.place(relx=.5,rely=.2, relwidth=.9,anchor='c')


b1=tk.Button(background,text='switch',font=font(size=12),command=switch)
b1.place(relx=.47,rely=.1, anchor='c')

f1_2=tk.Frame(root,bg='white')
f1_2.place(relx=.65,relwidth=.35,relheight=1)

l2 = tk.Label(f1_2, text='Fahrenheit', font=font(), background='white')
l2.place(relx=.5, rely=.1,anchor='c')

fahrenheit=tk.StringVar()
e2 = tk.Label(f1_2, font=font(), bg='lightgray',textvariable=fahrenheit)
e2.place(relx=.5, rely=.2, relwidth=.9,anchor='c')

# frame 2
f2=tk.Frame(root,bg='white')
f2.place(relwidth=.35,relheight=1)

l1_2 = tk.Label(f2, text='Fahrenheit', font=font(), background='white')
l1_2.place(relx=.5, rely=.1,anchor='c')


l_input.trace_add('write',ftoC)
e1_2 = tk.Entry(f2, font=font(), bg='lightgray', textvariable=l_input)
e1_2.place(relx=.5,rely=.2, relwidth=.9,anchor='c')

f2_2=tk.Frame(root,bg='white')
f2_2.place(relx=.65,relwidth=.35,relheight=1)

l2_2 = tk.Label(f2_2, text='Celsius', font=font(), background='white')
l2_2.place(relx=.5, rely=.1,anchor='c')
```

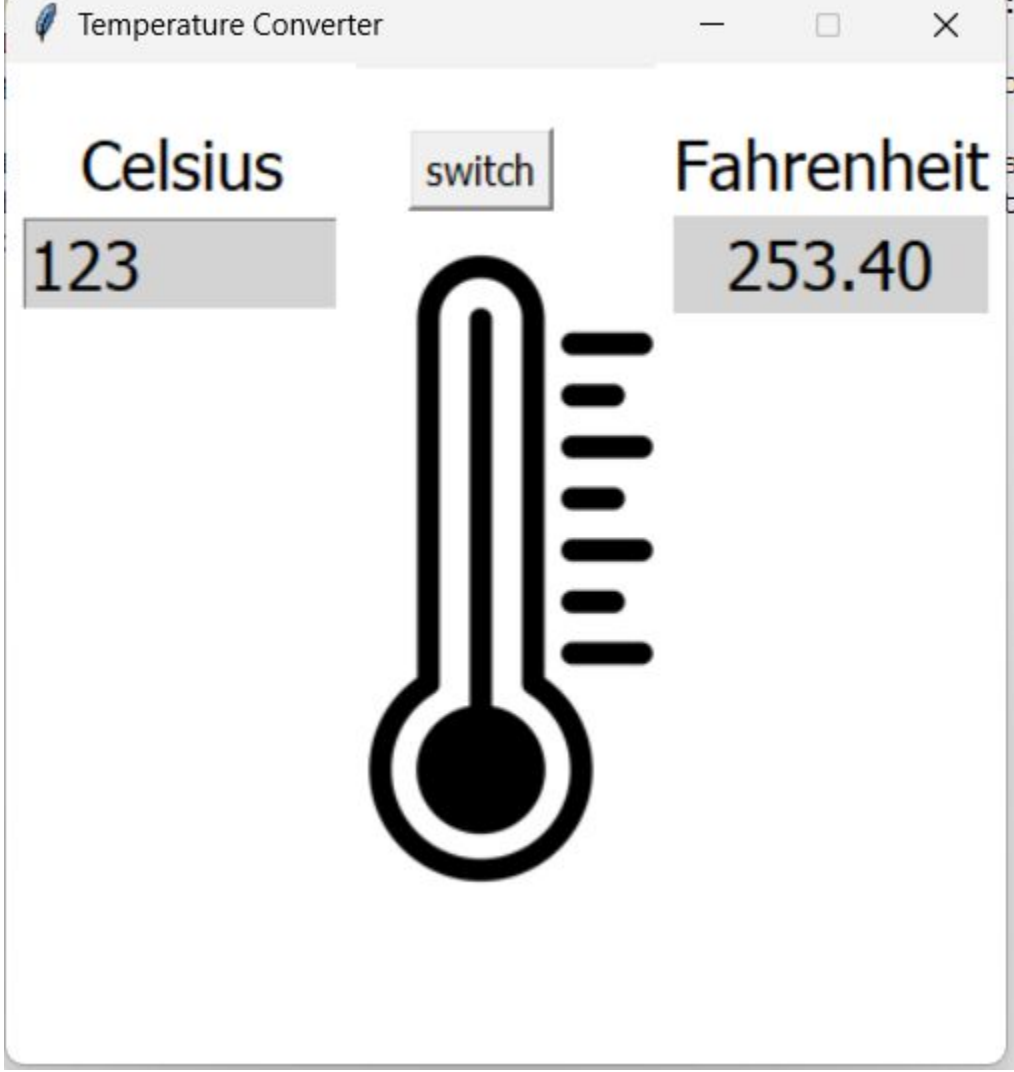
back

Ch2-bonus-ex01 code 3

```
celsius2=tk.StringVar()
e2_2 = tk.Label(f2_2, font=font(), bg='lightgray',textvariable=celsius2)
e2_2.place(relx=.5, rely=.2, relwidth=.9,anchor='c')


first_frames=[f1,f1_2]
second_frames=[f2,f2_2]
for i in first_frames:
    i.tkraise()
# tk.Button(root, text='Button', font=font).place(relx=.5, rely=.5, anchor='c')
# start app
root.mainloop()
```

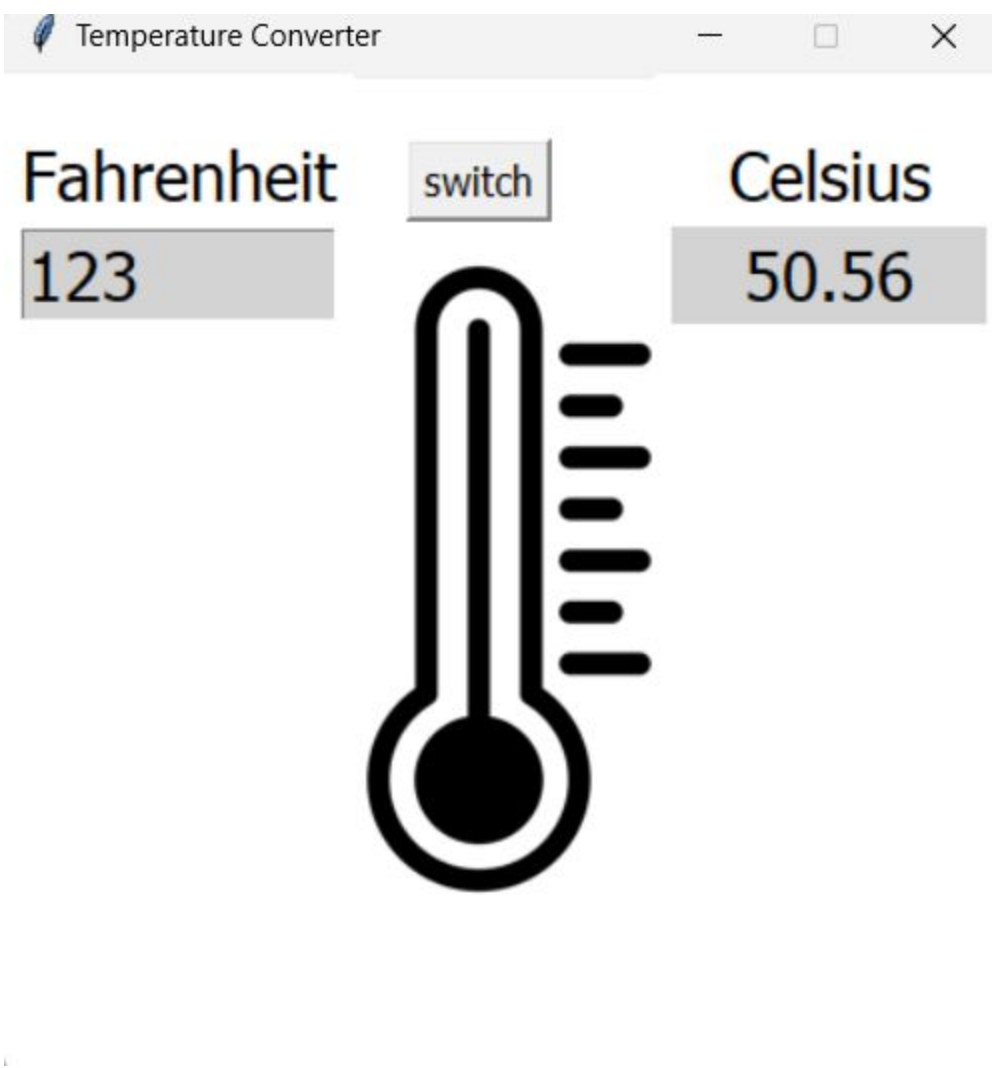
back

Ch2-bonus-ex01 output 1



back

Ch2-bonus-ex01 output 2



back

Ch2-bonus-ex02 code 1

```
import tkinter as tk
from PIL import ImageTk,Image
import datetime
root=tk.Tk()
font=('Tahoma',16)
date= tk.StringVar()
month=tk.StringVar()
year=tk.StringVar()


def message(*args):
    global date,month,year,l4
    test=[len(date.get()),len(month.get()),len(year.get())]
    if 0 in test:
        pass
    else:
        today=datetime.datetime.now()

        t_md=(f'{today.month}',f'{today.day}')
        b_md=(month.get(),date.get())

        age = int(today.year)-int(year.get())
        if str(t_md) < str(b_md):
            age-=1

        # remove any previously-defined l4 widget. This is because we are creating a
        # Everytime this code block runs (inside 'else'), it removes an l4 that coul
        l4.destroy()
        l4=tk.Label(root,font=font,bg='white',text=f'You are {age} years old')
        l4.place(relx=.5,rely=.4,anchor='c')



root.title('Age calculator')
root.geometry('400x400')
base_img=Image.open('Images/cake-logo-vector-25934828.jpg')
resize_img = base_img.resize((400,400))
final_img= ImageTk.PhotoImage(resize_img)

bg=tk.Label(root,image=final_img)
bg.place(x=0)

l1=tk.Label(bg,font=font,text='Date',bg='white')
l1.place(relx=.1,rely=.1)


e1=tk.Entry(root,font=font,textvariable=date,bg='lightgrey')
```

back

Ch2-bonus-ex02 code 2

```
e1=tk.Entry(root,font=font,textvariable=date,bg='lightgrey')
e1.place(relx=.07,rely=.2,relwidth=.2)
date.trace_add('write',message)

l2=tk.Label(bg,font=font,text='Month',bg='white')
l2.place(relx=.4,rely=.1)

e2=tk.Entry(root,font=font,textvariable=month,bg='lightgrey')
e2.place(relx=.39,rely=.2,relwidth=.2)
month.trace_add('write',message)

l3=tk.Label(bg,font=font,text='Year',bg='white')
l3.place(relx=.7,rely=.1)

e3=tk.Entry(root,font=font,textvariable=year,bg='lightgrey')
e3.place(relx=.67,rely=.2,relwidth=.2)
year.trace_add('write',message)

# define l4 to be so the function runs properly
l4=tk.Label(root)


root.mainloop()
```

back

Ch2-bonus-ex02 output

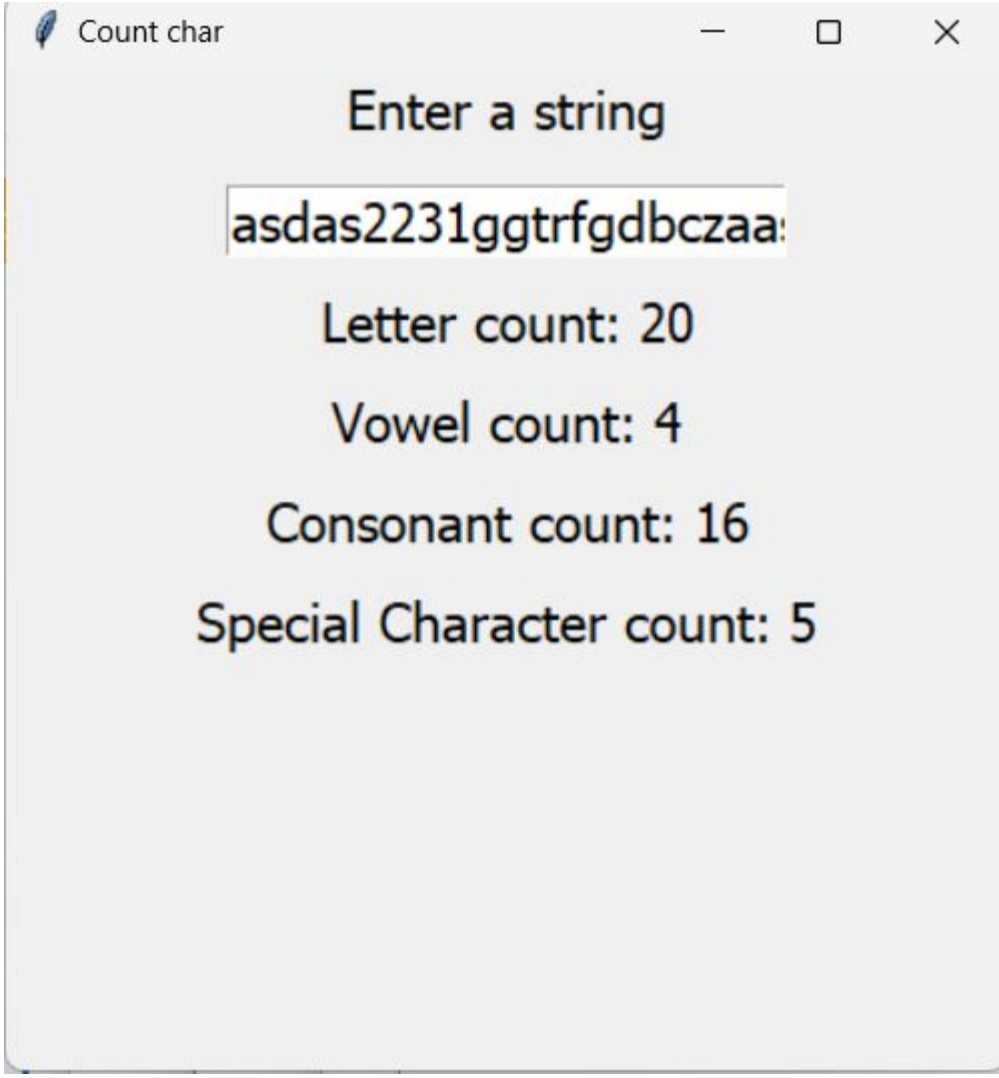
back

Ch2-further-ex01 code

```
import tkinter as tk

root=tk.Tk()
font=('Tahoma',16)
def output(*args):

    global l_list, font
    alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
    vowels = ['a', 'e', 'i', 'o', 'u']
    l_count=0
    v_count=0
    c_count=0
    s_count=0
    for i in string_id.get():
        char= i.lower()
        if char in alphabet:
            l_count+=1
            if char in vowels:
                v_count+=1
            elif char not in vowels:
                c_count+=1

        else:
            s_count+=1

    counts=[l_count,v_count,c_count,s_count]
    string_n=['Letter','Vowel','Consonant','Special Character']
    y=[.25,.35,.45,.55]
    increment=0
    for i in l_list:
        i.destroy()
        i=tk.Label(root,text=f'{string_n[increment]} count: {counts[increment]}',font=font)
        i.place(relx=.5,rely=y[increment],anchor='c')
        increment+=1


root.title('Count char')
root.geometry('400x400')


tk.Label(root,text='Enter a string',font=font).place(relx=.5,anchor='n')

string_id=tk.StringVar()

e1=tk.Entry(root,textvariable=string_id,font=font)
e1.place(relx=.5,rely=.15,anchor='c')
string_id.trace_add('write',output)
```

back

Ch2-further-ex01 output



back

Ch2-further-ex02 code 1

```
import tkinter as tk

root=tk.Tk()
font=('Tahoma',16)
def capitalize_string(*strings):
    global labels
    chars=string.get() # get the user input
    max_chars=30 # arbitrary max characters

    # list that stores an element up to 33 in length
    wrapper=[chars[i:i+max_chars] for i in range(0,len(chars),max_chars)]

    # destroy every object per  call so that that the labels update in real-time (backspace
    for i in labels:

        i.destroy()

    increment=0
    y=.2
    for i in wrapper:
        # since the previous labels are 'destroyed' on the app (list unaffected), we have to
        #Referencing elements from 'labels' is just so that we have objects to point to when
        labels[increment]=tk.Label(root,font=font,text=i.upper())
        labels[increment].place(relx=.5,rely=y,anchor='n')
        increment+=1
        y+=.1

root.geometry('400x400')
root.title('Capitalize Letters')

tk.Label(root,text='Enter string:',font=font).place(relx=.5,anchor='n')

string=tk.StringVar()
string.trace_add('write',capitalize_string)
tk.Entry(root,textvariable=string,font=font).place(relx=.5,rely=.1,anchor='n')

l1=tk.Label(root,font=font)
l2=tk.Label(root,font=font)
l3=tk.Label(root,font=font)
l4=tk.Label(root,font=font)
l5=tk.Label(root,font=font)
l6=tk.Label(root,font=font)
l7=tk.Label(root,font=font)
l8=tk.Label(root,font=font)
l9=tk.Label(root,font=font)
l10=tk.Label(root,font=font)
```

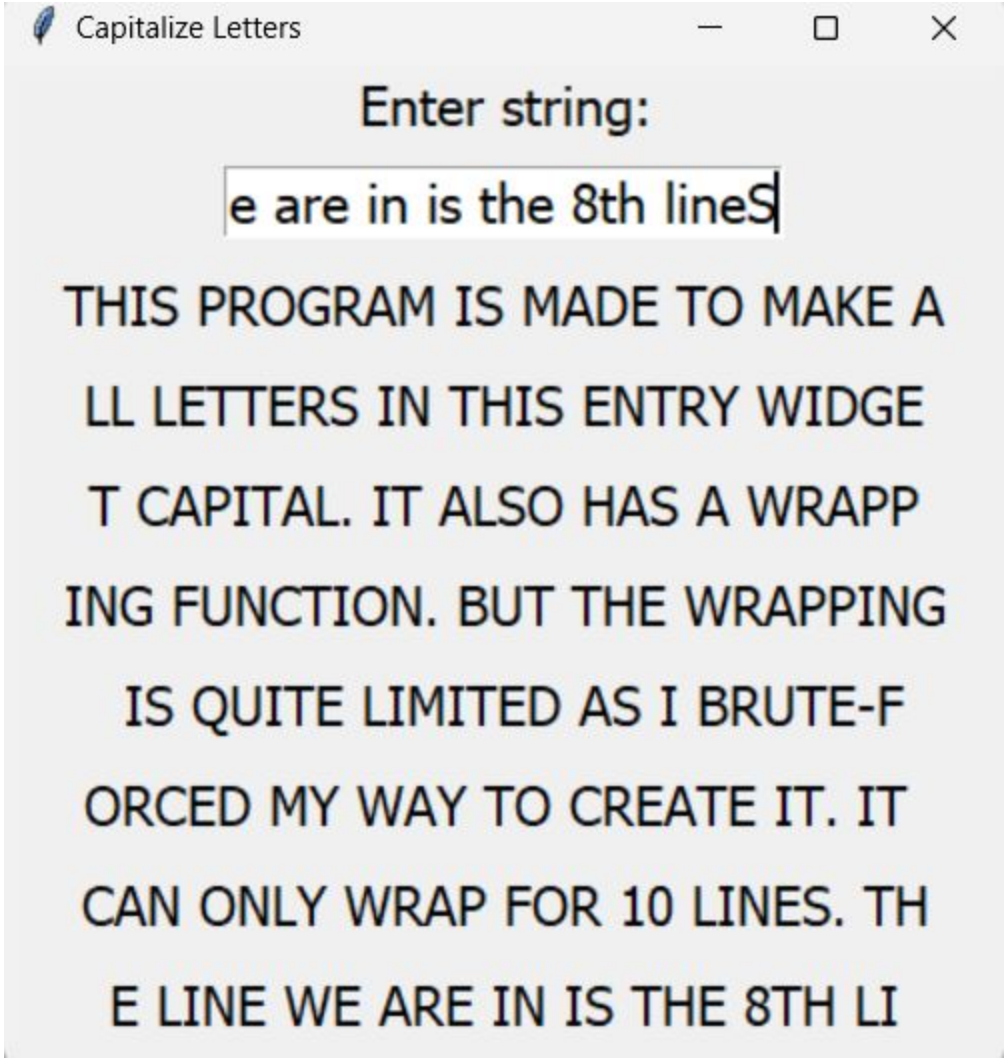
back

Ch2-further-ex02 code 2

```
# these need to be objects because we will destroy
labels=[l1,l2,l3,l4,l5,l6,l7,l8,l9,l10]
```

back

Ch2-further-ex02 output



back

Ch3-ex01 code

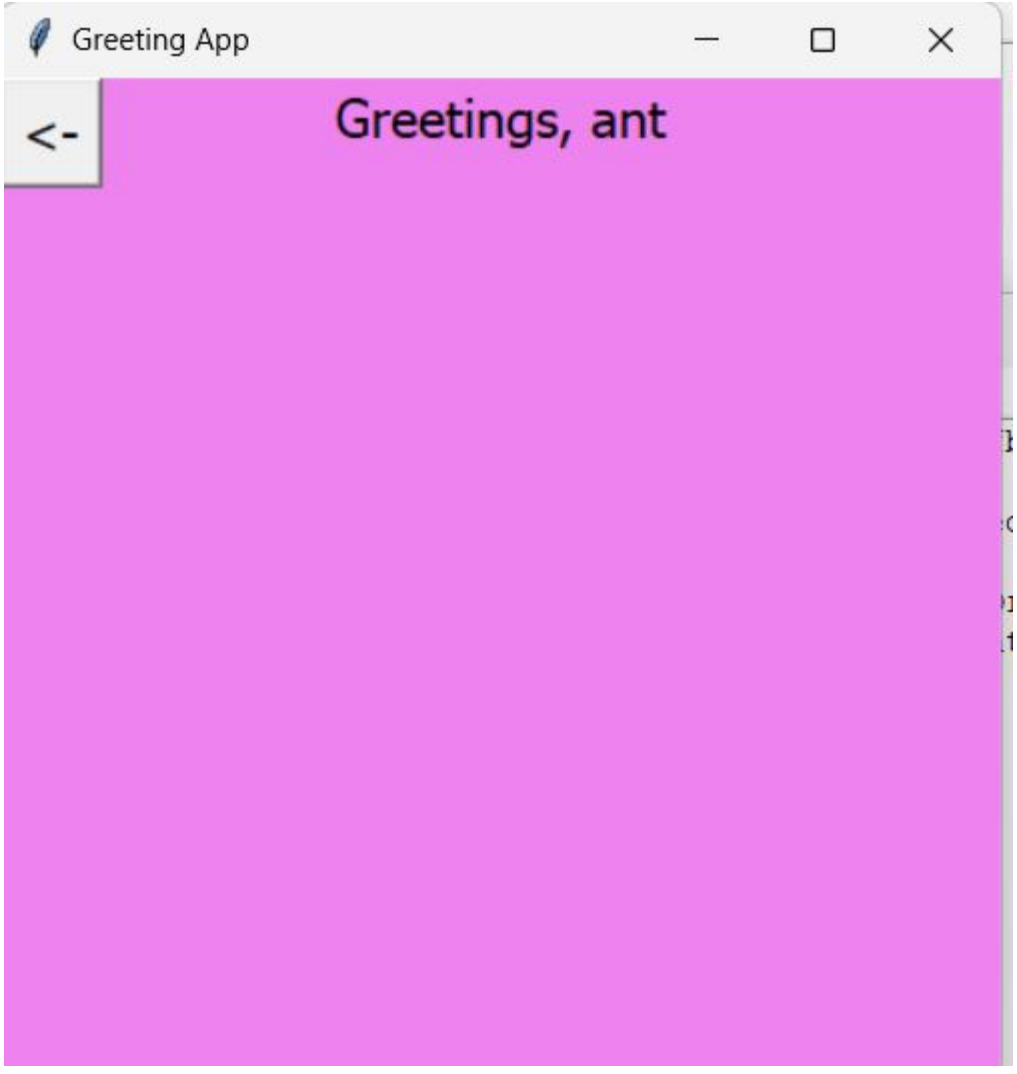
```
import tkinter as tk

root=tk.Tk()
font=('Tahoma',16)
name=tk.StringVar()
color=tk.StringVar()

def switch():
    global frames
    DisplayFrame.configure(bg=color.get())
    l1.configure(text=f'Greetings, {name.get()}',bg=color.get())
    frames[-1].tkraise()
    frames[0],frames[-1]=frames[-1],frames[0]


root.geometry('400x400')
root.title('Greeting App')

InputFrame=tk.Frame(root,bg='green')
InputFrame.place(relwidth=1,relheight=1)

tk.Label(InputFrame,text='Greeting app',font=font,fg='blue',bg='green').place(relx=.5,anchor='n')
tk.Entry(InputFrame,textvariable=name,font=font).place(relx=.5,rely=.1,anchor='n')

color.set('Orange')
tk.OptionMenu(InputFrame,color,'Orange','Violet','Red').place(relx=.5,rely=.2,anchor='n')

tk.Button(InputFrame,text='Update Greeting',font=font,fg='blue',bg='green',command=switch).place(relx=.5,rely=.3,anchor='n')


DisplayFrame=tk.Frame(root,bg=color.get())
DisplayFrame.place(relwidth=1,relheight=1)

l1=tk.Label(DisplayFrame,font=font,bg=color.get())
l1.place(relx=.5,anchor='n')

tk.Button(DisplayFrame,text='<-',font=font,command=switch).place(relx=0)
InputFrame.tkraise()
frames=[InputFrame,DisplayFrame]
```

back

Ch3-ex01 output 1

back

Ch3-ex01 output 2



back

Ch3-ex02 code 1

```
import tkinter as tk
from tkinter import ttk,messagebox
from PIL import ImageTk,Image
root=tk.Tk()

# All of these vars are global because they will be present in more than 1 function. Or they're reused.
# because I don't like writing the tuple again and again
font=('Tahoma',16)

# variable to hold values of the sliders
sugar_v=tk.IntVar()
milk_v=tk.IntVar()

deposit_v=tk.IntVar()

# user's money. global because we need a counter for the amount of money the user has deposited, and it has to interact with different functions
coin=0

coffee=['americano','black','cappucino','espresso','latte','irish']

# list of coffee images. global to prevent garbage collection and since we'll use it again
l=[]

# counter for cost. global because it interacts with at least 2 functions
cost=0

# gatekeep certain function codes unless set to True
select_coffee=False

# Snapshot price of coffee only
c_cost=0

# tracks the index value of the pressed coffee
tracker=0


def create_coffees():
    global coffee,l
    x=0
    y=0
    increment=0
    for i in coffee:
        base=Image.open(f'C:/Users/MICRO/OneDrive/Documents/GitHub/skills-portfolio-Antonio-III/Chapter 3 - Graphical User Interface(Cont.)/Images/{i}.jpg')
        resize=base.resize((110,140))
        img=ImageTk.PhotoImage(resize)
        l.append(img)
        button=tk.Button(root,image=l[increment], command= lambda index=increment: on_select_coffee(index),bg='red')
```

back

Ch3-ex02 code 2

```
        img=ImageTk.PhotoImage(resize)
        l.append(img)
        button=tk.Button(root,image=l[increment], command= lambda index=increment: on_select_coffee(index),bg='red')
        # prevent garbace collection by a var attaching to an object. This is because objects don't get their space in memory
        # button.dont_garbagecollect=l[increment] . REMOVED BECAUSE 'l' is now global

        button.place(x=x,y=y)
        x+=115
        increment+=1
        if i==coffee[2]:
            x=0
            y+=145

def on_slide(event):
    global select_coffee,sugar_v,milk_v,sugar_c, milk_c,cost,coffee_c,c_cost

    if select_coffee==True:
        # get sugar and milk values
        sugar=sugar_v.get()
        milk=milk_v.get()
        # update slider values to reflect changes
        sugar_c.configure(text=f'Sugar: {sugar}')
        milk_c.configure(text=f'Milk:  {milk}')

        # update cost. Disregard the previous 'cost' amount when using sliders because using it here will make the price stac
        cost=sugar+(milk*2)+c_cost
        # update 'cost' text to reflect changes
        coffee_c.configure(text=f'Cost: ${cost}')

    else:
        messagebox.showerror('Error','No coffee selected')

def on_slide_c(event):
    global deposit_c,deposit_v
    # update values
    deposit_c.configure(text=deposit_v.get())

def deposit_coin():
    global deposit_v,coin,usable_coin

    added_coin=deposit_v.get()
    # increase money count
    coin+=int(added_coin)
    # update text to reflect changes
    usable_coin.configure(text=f'Coins: ${coin}')

def on_select_coffee(coffee):
```

back

Ch3-ex02 code 3

```
def on_select_coffee(coffee):
    global l, c_cost, select_coffee,tracker,cost

    # Show display item
    selected_coffee.configure(image=l[coffee])

    # This var serves as a snapshot of the coffee cost itself
    c_cost= 0+coffee+1
    # 'cost' will update
    cost=c_cost

    # reflect changes
    coffee_c.configure(text=f'Cost: ${cost}')

    # condition to track if the user has called this function (ie pressed one of the coffees)
    select_coffee=True

    # pass a value that represents one of the index position of 'l'. This is so that a function-defined var can become global
    tracker=coffee

def buy_coffee():
    global select_coffee,coin,cost,tracker,usable_coin,sugar_v,milk_v
    sugar=sugar_v.get()
    milk=milk_v.get()
    if select_coffee==True:
        if coin>=cost:
            # transaction
            coin-=cost
            # transaction success
            messagebox.showinfo('Success',f'You bought: {coffee[tracker].title()} Coffee. With {sugar} spoon of sugar and {milk} spoon of milk. ${cost} was deducted to your balance.')
            # update coin GUI
            usable_coin.configure(text=f'Coins: {coin}')
        else:
           messagebox.showerror('Error','Not enough funds')

root.geometry('550x450')
root.title('Cofee Vending Machine')
root.resizable(0,0)
# frame (vending machine bg)
frame=tk.Frame(root,bg='red')
frame.place(width=350,relheight=1)

# coffee items
create_coffees()

# selected
tk.Label(root,text='Selected:',font=font).place(x=390)
```

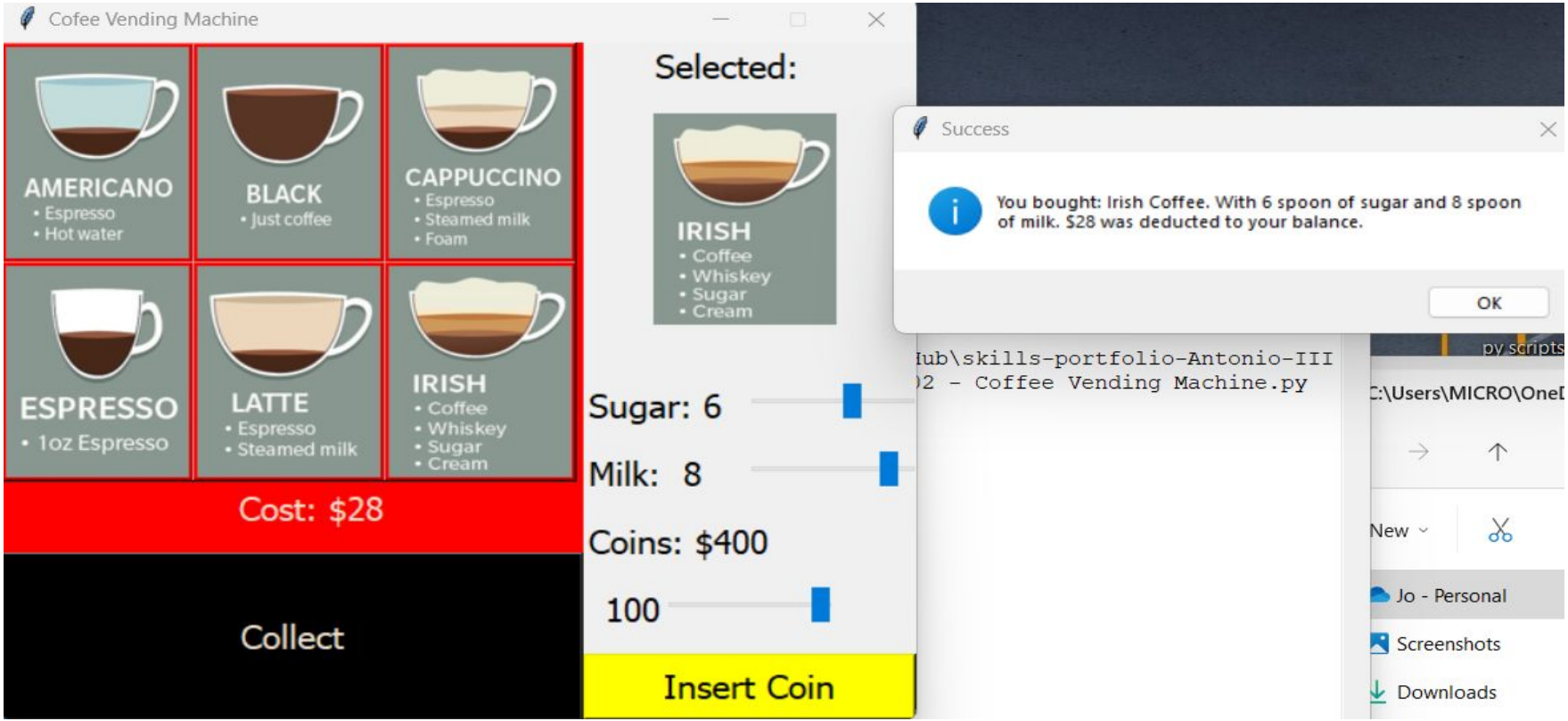
back

Ch3-ex02 code 4

```
# selected
tk.Label(root,text='Selected:',font=font).place(x=390)
selected_coffee=tk.Label(root)
selected_coffee.place(x=390,rely=.1)

# options (right-side)
sugar_c=tk.Label(root,text=f'Sugar: {sugar_v.get()}',font=font)
sugar_c.place(x=350,rely=.5)

ttk.Scale(root,from_=0, to=10,variable=sugar_v,command=on_slide).place(x=450,rely=.5)

milk_c=tk.Label(root,text=f'Milk:  {milk_v.get()}',font=font)
milk_c.place(x=350,rely=.6)


ttk.Scale(root,from_=0, to=10,variable=milk_v,command=on_slide).place(x=450,rely=.6)


# currency
# coins:0
usable_coin= tk.Label(root,text=f'Coins: ${coin}',font=font)
usable_coin.place(x=350,rely=.7)

# 0 on left of slider
deposit_c=tk.Label(root,text=deposit_v.get(),font=font)
deposit_c.place(x=360,rely=.80)

# slider
ttk.Scale(root,from_=0,to=100,variable=deposit_v,command =on_slide_c).place(x=400,rely=.8)

tk.Button(root,text='Insert Coin',font=font,command=deposit_coin,bg='yellow',fg='black').place(x=350,rely=.9,width=200)

# collect coffee
coffee_c=tk.Label(frame,text=f'Cost: ${cost}',font=font,bg='red',fg='antiquewhite2')
coffee_c.place(relx=.4,rely=.65)

tk.Button(frame,text='Collect',font=font,command=buy_coffee,bg='black',fg='antiquewhite2').place(x=0,rely=.75,relwidth=1,relheight=.25)


root.mainloop()
```

back

Ch3-ex02 output



back

Ch3-ex03 code 1

```
import tkinter as tk
import math
root=tk.Tk()

# VARS
font=('Tahoma',16)

# radius
string_vr=tk.StringVar()

# side
string_vs=tk.StringVar()

# width
string_vw=tk.StringVar()
# length
string_vl=tk.StringVar()

# frames.place() preset
f_place_p={'relx':0,'rely':.1,'relwidth':1,'relheight':.9}
```

back

Ch3-ex03 code 2

```
def switch_tabs(target_tab):
    global shapes_f,formula,canvas
    shapes_f[target_tab].tkraise()



def getr(*args):
    global string_vr
    input_r=(string_vr.get())
    if ' ' in input_r or len(input_r)==0:
        input_rint=0
    else:
        input_rint=int(input_r)


    calculation=math.pi*input_rint**2
    formula_c.configure(text=f'A ≈ {calculation}')

def gets(*args):
    global string_vs
    input_s=(string_vs.get())
    if ' ' in input_s or len(input_s)==0:
        input_sint=0
    else:
        input_sint=int(input_s)

    calculation=input_sint**2

    formula_s.configure(text=f'A = {calculation}')

def getwl(*args):
    width=string_vw.get()
    length=string_vl.get()
    wl=[width,length]
    inc=0
    usable=bool
    for i in wl:
        if len(wl[inc])==0 or ' ' in wl[inc]:
            usable=False
        inc+=1

    if usable !=False:
        w_int=int(width)
        l_int=int(length)

        calculation=w_int*l_int
        formula_r.configure(text=f'A = {calculation}')
```

back

Ch3-ex03 code 3

```
# header
root.title('Area Function')
root.geometry('500x500')


# (root,from_,to,bg,fg,font,text,command)
# main
tabs=tk.Frame(root,bg='red')
tabs.place(relwidth=1,relheight=.1)

tk.Button(tabs,font=font,text='circle',command=lambda: switch_tabs(0)).place(relwidth=.2,relheight=1)
tk.Button(tabs,font=font,text='square',command=lambda: switch_tabs(1)).place(relx=.2,relwidth=.2,relheight=1)
tk.Button(tabs,font=font,text='rectangle',command=lambda: switch_tabs(2)).place(relx=.4,relwidth=.2,relheight=1)

# color bg of every tab: aqua, lightgoldenrod1, chartreuse

# circle tab
circle_f=tk.Frame(root,bg='aqua')
circle_f.place(**f_place_p)

# formula
formula_c=tk.Label(circle_f,bg='aqua',font=font,text='A = π r^2')
formula_c.place(relx=.5,anchor='n')

# canvas
canvas=tk.Canvas(circle_f,bg='aqua',highlightthickness=0)
canvas.place(relx=.5,rely=.4,width=300,height=300,anchor='c')

#circle and diameter line
circle=canvas.create_oval(10,10,290,290,outline='darkorchid2',width=5)
diameter_l=canvas.create_line(10,150,290,150,fill='darkorchid2',width=5)
# center dot and 'd'
#center_d=canvas.create_oval(140,145,150,155,fill='black')

#letter_d=tk.Label(canvas,font=font,bg='aqua',fg='black',text='d')
#letter_d.place(relx=.5,rely=.55)

# radius line, 'r',
radius_l=canvas.create_line(150,145,235,45,fill='crimson',width=5)

letter_r=tk.Label(canvas,font=font,bg='aqua',fg='crimson',text='r')
letter_r.place(relx=.6,rely=.29)

# 'radius' and entry
radius_t=tk.Label(circle_f,font=font,bg='aqua',fg='crimson',text='radius')
radius_t.place(relx=.2,rely=.8)
```

back

Ch3-ex03 code 4

```
# 'radius' and entry
radius_t=tk.Label(circle_f,font=font,bg='aqua',fg='crimson',text='radius')
radius_t.place(relx=.2,rely=.8)

entry_c=tk.Entry(circle_f,font=font,bg='aqua',fg='crimson',highlightthickness=1,highlightbackground='grey',textvariable=string_vr)
entry_c.place(relx=.35,rely=.8)
string_vr.trace_add('write',getr)
#---

# square tab
square_f=tk.Frame(root,bg='lightgoldenrod1')
square_f.place(**f_place_p)

# formula
formula_s=tk.Label(square_f,bg='lightgoldenrod1',font=font,text='A = s^2')
formula_s.place(relx=.5,anchor='n')

#canvas
canvas_s=tk.Canvas(square_f,bg='lightgoldenrod1',highlightthickness=0)
canvas_s.place(relx=.5,rely=.4,width=300,height=300,anchor='c')

# square shape
square=canvas_s.create_rectangle(10,10,290,290,outline='cornflowerblue',width=5)

# s label
letter_s=tk.Label(canvas_s,font=font,bg='lightgoldenrod1',fg='cornflowerblue',text='s')
letter_s.place(relx=.94,rely=.45)

# 'side' and entry
side_t=tk.Label(square_f,font=font,bg='lightgoldenrod1',fg='cornflowerblue',text='side')
side_t.place(relx=.2,rely=.8)

entry_s=tk.Entry(square_f,font=font,bg='lightgoldenrod1',fg='cornflowerblue',highlightthickness=1,highlightbackground='grey',textvariable=string_vs)
entry_s.place(relx=.35,rely=.8)
string_vs.trace_add('write',gets)
#---
# rectangle tab
rectangle_f=tk.Frame(root,bg='chartreuse')
rectangle_f.place(**f_place_p)

# formula
formula_r=tk.Label(rectangle_f,bg='chartreuse',font=font,text='A = wl')
formula_r.place(relx=.5,anchor='n')

#canvas
canvas_r=tk.Canvas(rectangle_f,bg='chartreuse',highlightthickness=0)
```

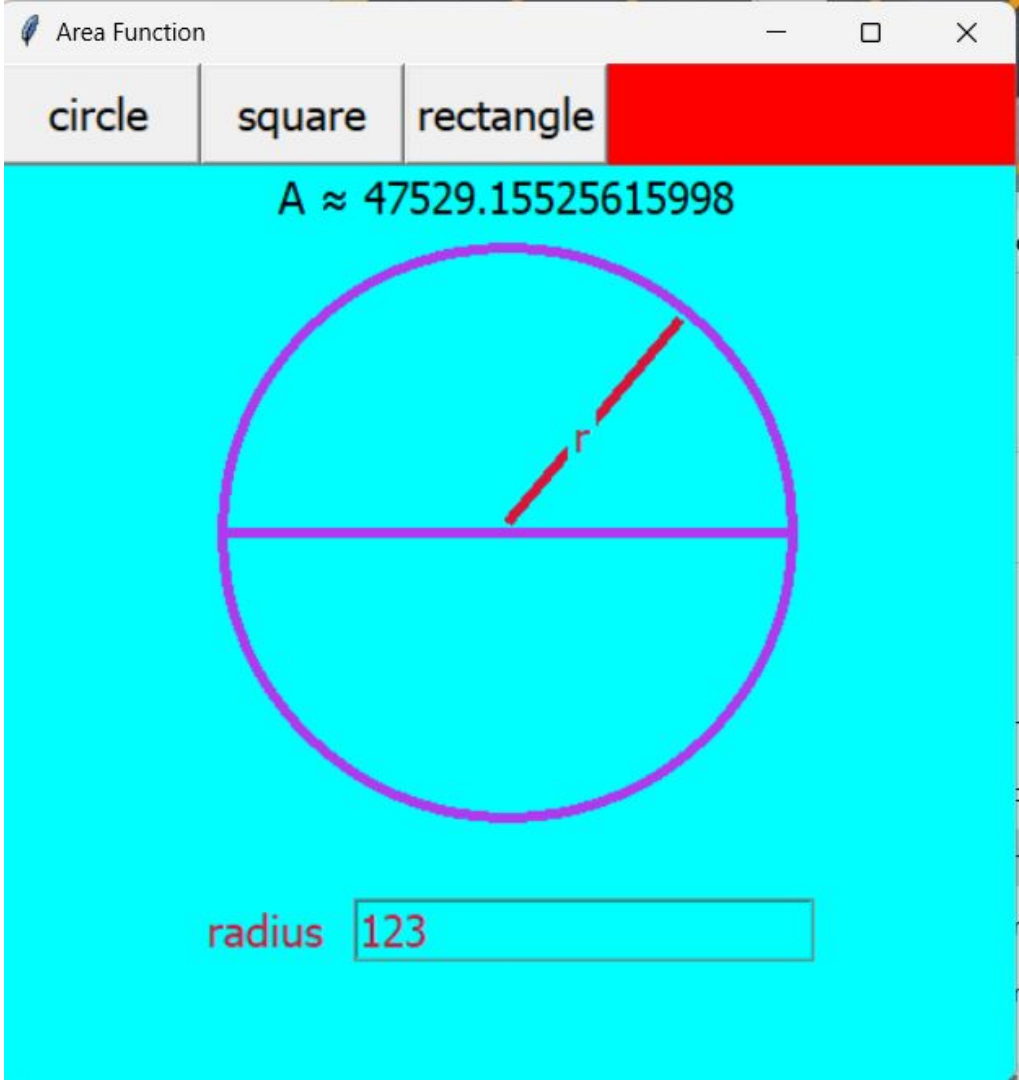
back

Ch3-ex03 code 5

```
#canvas
canvas_r=tk.Canvas(rectangle_f,bg='chartreuse',highlightthickness=0)
canvas_r.place(relx=.5,rely=.4,width=500,height=300,anchor='c')

# rectangle shape
rectangle=canvas_r.create_rectangle(50,50,450,250,outline='red',width=5)

# w and l label
letter_w=tk.Label(rectangle_f,font=font,bg='chartreuse',fg='red',text='w')
letter_w.place(relx=.08,rely=.37)

letter_l=tk.Label(rectangle_f,font=font,bg='chartreuse',fg='red',text='l')
letter_l.place(relx=.5,rely=.59)

# 'width' and 'length' text

width_t=tk.Label(rectangle_f,font=font,bg='chartreuse',fg='red',text='width')
width_t.place(relx=.2,rely=.7)

length_t=tk.Label(rectangle_f,font=font,bg='chartreuse',fg='red',text='length')
length_t.place(relx=.2,rely=.8)
# w and l entry

entry_rw=tk.Entry(rectangle_f,font=font,bg='chartreuse',fg='red',textvariable=string_vw)
entry_rw.place(relx=.35,rely=.7)
string_vw.trace_add('write',getwl)

entry_rl=tk.Entry(rectangle_f,font=font,bg='chartreuse',fg='red',textvariable=string_vl)
entry_rl.place(relx=.35,rely=.8)
string_vl.trace_add('write',getwl)
#---
shapes_f=[circle_f,square_f,rectangle_f]
shapes_f[0].tkraise()
root.mainloop()
```

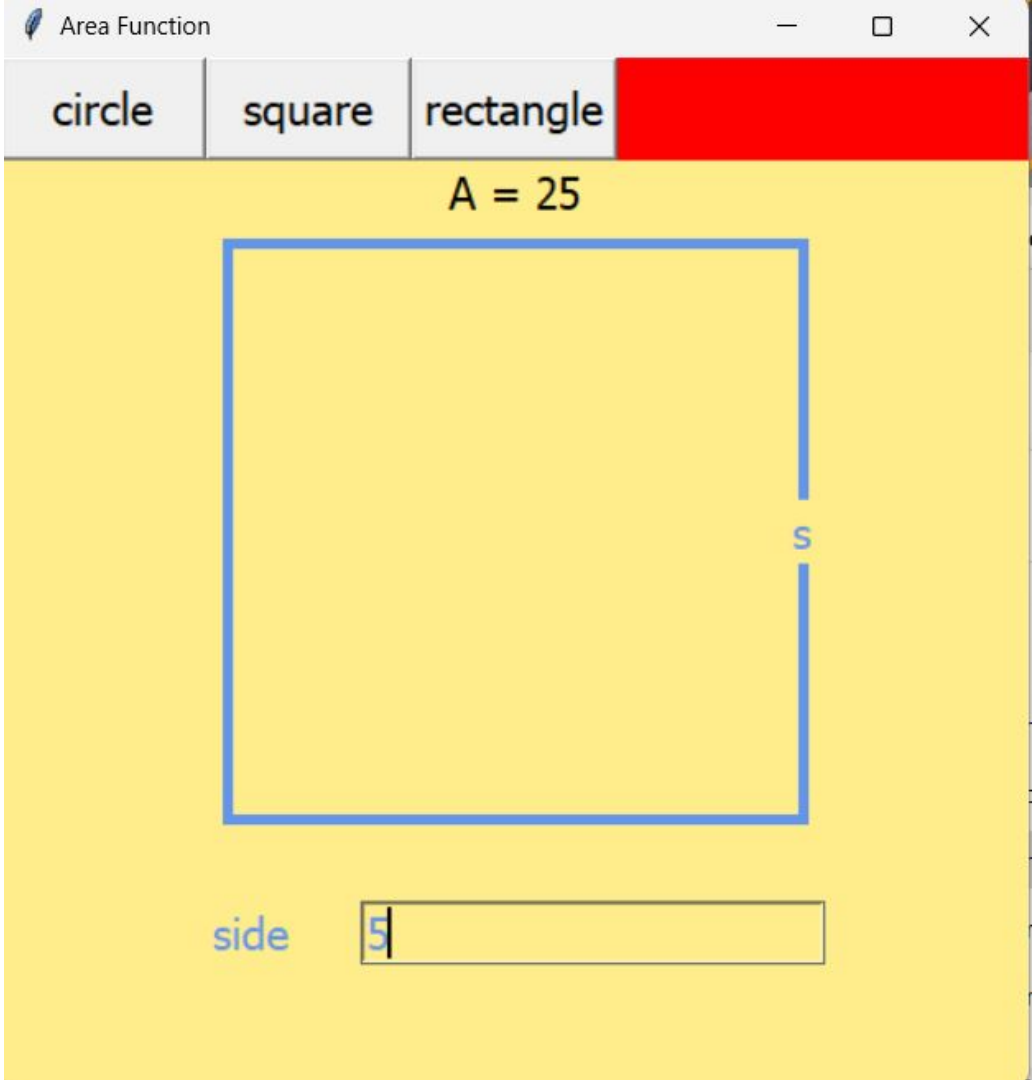
back

Ch3-ex03 output 1



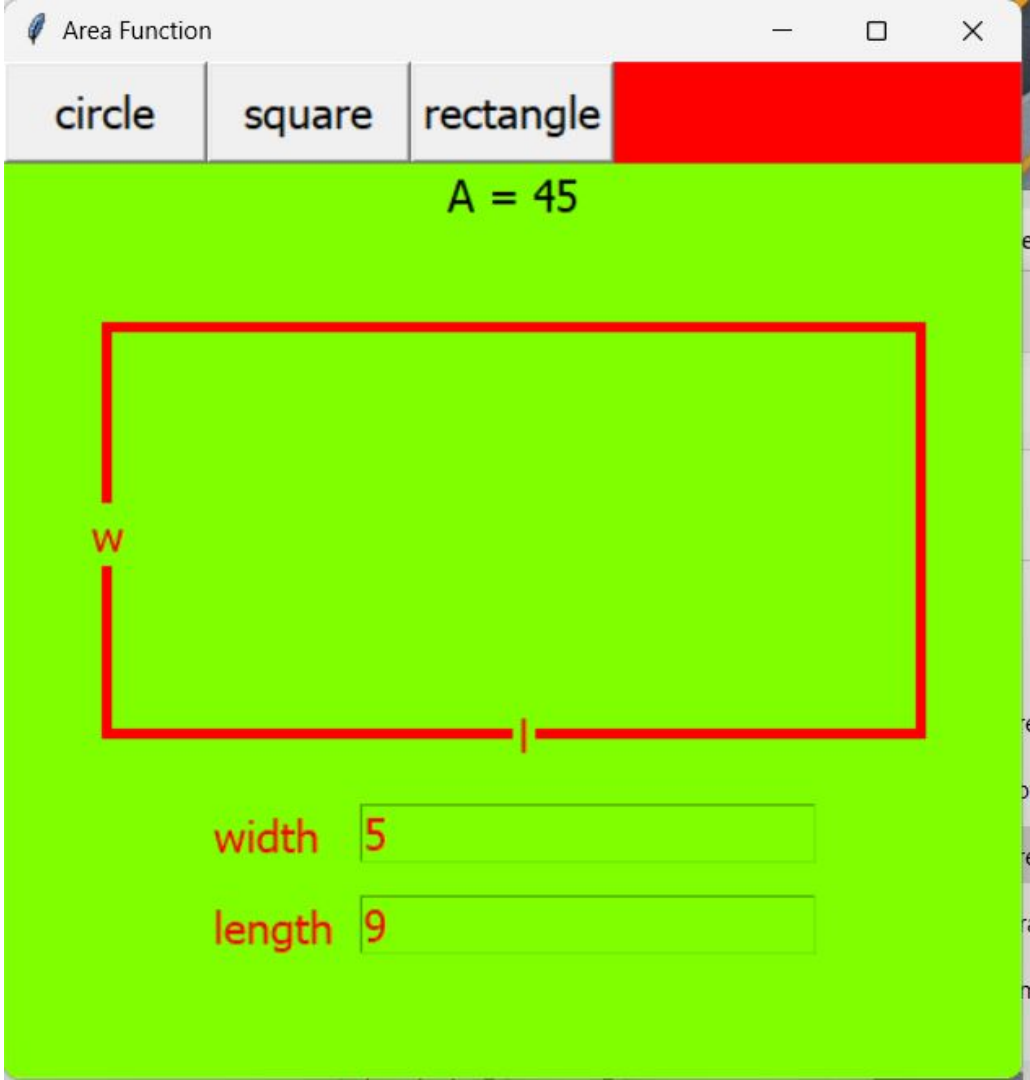
back

Ch3-ex03 output 2



back

Ch3-ex03 output 3



back

Ch3-ex04 code 1

```
import tkinter as tk

root=tk.Tk()
def place_shape(mousepos):
    global activeshape,canvas,created_shapes
    x,y=mousepos.x,mousepos.y

    if active_shape=='oval':
        c_shape= canvas.create_oval(x,y,x+70,y+50,fill='red',outline='red',width=5)
    elif active_shape=='rectangle':
        c_shape=canvas.create_rectangle(x,y,x+70,y+50,fill='blue',outline='blue',width=5)
    elif active_shape=='square':
        c_shape=canvas.create_rectangle(x,y,x+50,y+50,fill='green',outline='green',width=5)
    elif active_shape=='triangle':
        points=[x,y,x-20,y+40,x+20,y+40]
        c_shape=canvas.create_polygon(points,fill='yellow',outline='yellow',width=5)

    created_shapes.append(c_shape)

def change_shape(new_shape):
    global active_shape
    active_shape=new_shape

def clear(*args):
    global created_shapes

    for i in created_shapes:
        canvas.delete(i)

    if len(created_shapes)>0:
        created_shapes.clear()
font=('Tahoma',16)
root.title('Draw Shape')
root.geometry('400x400')


options=tk.Frame(root,bg='white')
options.place(relwidth=1,relheight=.1)

# oval, rectangle, square, triangle
# (root,from_,to,bg,fg,font,text,command)
oval_b=tk.Button(options,font=font,text='oval',command=lambda:change_shape('oval'))
oval_b.place(relwidth=.25)

rectangle_b=tk.Button(options,font=font,text='rectangle',command=lambda:change_shape('rectangle'))
rectangle_b.place(relx=.25,relwidth=.25)
```

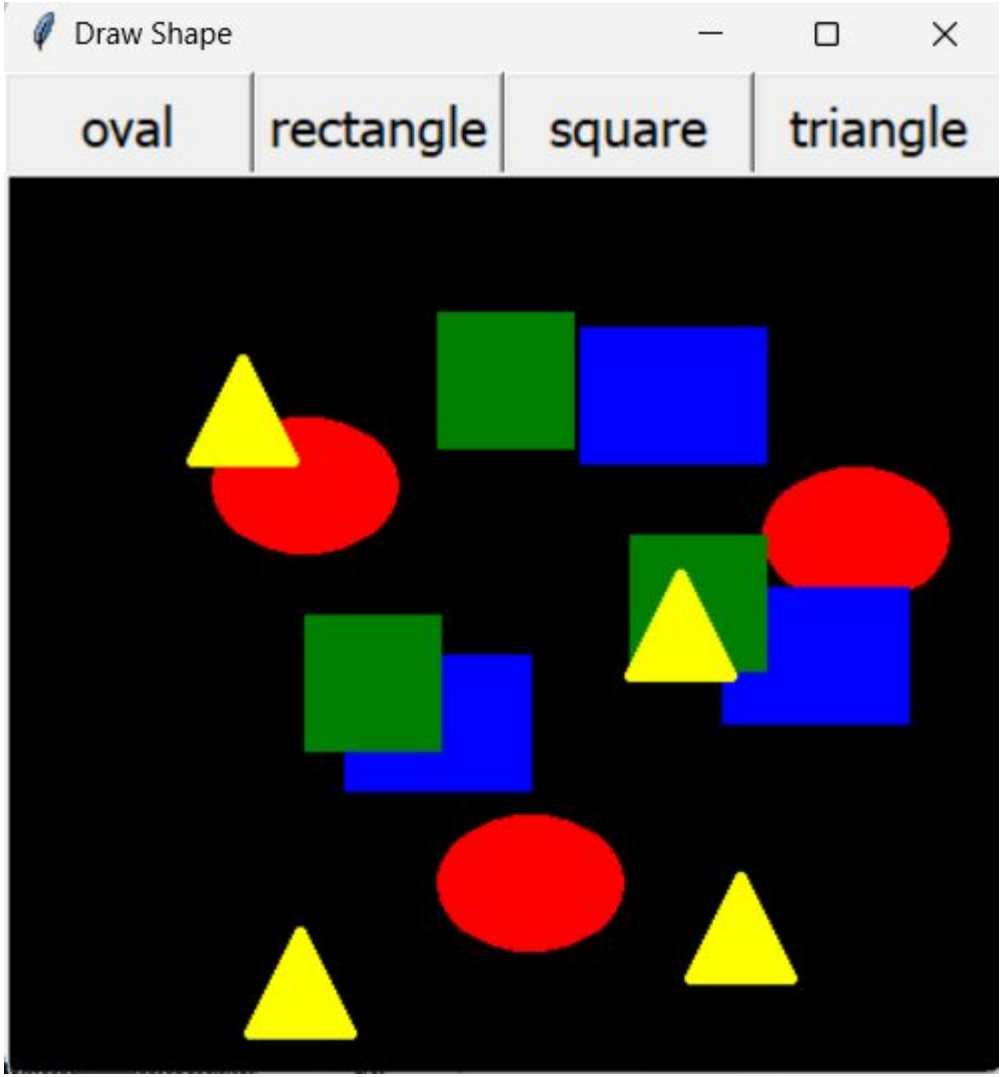
back

Ch3-ex04 code 2

```
rectangle_b=tk.Button(options,font=font,text='rectangle',command=lambda:change_shape('rectangle'))
rectangle_b.place(relx=.25,relwidth=.25)

square_b=tk.Button(options,font=font,text='square',command=lambda:change_shape('square'))
square_b.place(relx=.5,relwidth=.25)

triangle_b=tk.Button(options,font=font,text='triangle',command=lambda:change_shape('triangle'))
triangle_b.place(relx=.75,relwidth=.25)

canvas=tk.Canvas(root,bg='black')
canvas.place(y=40,relwidth=1,relheight=1)


canvas.bind('<Button-1>',place_shape)
root.bind('<Key-c>',clear)

active_shape='oval'
created_shapes=[]
root.mainloop()
```

back

Ch3-ex04 output



back

Ch4-ex01 code 1

```
import tkinter as tk

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title('Write to txt file')
        self.geometry('500x500')
        self['bg']='lightgoldenrod1'
        # vars
        font=('Tahoma',15)
        w_preset={'font':('Tahoma',15),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.file='Text Files/bio.txt'

        self.name_v=tk.StringVar()
        self.age_v=tk.StringVar()
        self.hometown_v=tk.StringVar()



        # widget rules (root,from_,to,bg,fg,font,text,command)

        # name
        tk.Label(self,**w_preset,text='Name: ').place(x=0)

        tk.Entry(self,font=font,textvariable=self.name_v).place(relx=.25)

        # age
        tk.Label(self,**w_preset,text='Age: ').place(rely=.1)

        tk.Entry(self,font=font,textvariable=self.age_v).place(relx=.25,rely=.1)

        # hometown
        tk.Label(self,**w_preset,text='Hometown: ').place(rely=.2)

        tk.Entry(self,font=font,textvariable=self.hometown_v).place(relx=.25,rely=.2)

        # button write
        tk.Button(self,**w_preset,text='Write to file',command=self.write).place(relx=.5,rely=.35,anchor='c')

        # button output
        tk.Button(self,**w_preset,text='Output file',command=self.output).place(relx=.5,rely=.45,anchor='c')

        self.txt_w=tk.Text(self,font=font)
        self.txt_w.place(relx=.5,rely=.55,relwidth=.9,relheight=.4,anchor='n')


    # functions
```

back

Ch4-ex01 code 2

```
    # functions
    def write(self):
        bio=self.file
        name=self.name_v.get()
        age=self.age_v.get()
        hometown=self.hometown_v.get()

        with open(bio,'w') as file_handler:
            file_handler.write(f'Name: {name}\n')
            file_handler.write(f'Age: {age}\n')
            file_handler.write(f'Hometown: {hometown}\n')


    def output(self):
        bio=self.file
        # get the text
        with open(bio) as file_handler:
            content=file_handler.read()
        #delete text first
        self.txt_w.delete('1.0','end')
        # insert
        self.txt_w.insert(tk.END,content)
if __name__=='__main__':
    App().mainloop()
```

back

Ch4-ex01 output

back

Ch4-ex02 code 1

```
class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Count String')
        self.geometry('500x500')
        self['bg']='lightgoldenrod1'

        # vars
        font=('Tahoma',16)
        self.w_preset=w_preset={'font':('Tahoma',15),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.file='Text Files/sentences.txt'
        self.search_v=tk.StringVar()
        self.target_strings=['Hello my name is Peter Parker','I love Python Programming','Love','Enemy']


        # (root,from_,to,bg,fg,font,text,command)
        tk.Label(self,**w_preset,text='Count the amount of times these strings occurred:').place(relx=.0)

        self.item_1=tk.Label(self,**w_preset)
        self.item_1.place(rely=.1)

        self.item_2=tk.Label(self,**w_preset)
        self.item_2.place(rely=.2)

        self.item_3=tk.Label(self,**w_preset)
        self.item_3.place(rely=.3)

        self.item_4=tk.Label(self,**w_preset)
        self.item_4.place(rely=.4)

        # execute function
        self.count_sentences()

        # Extension
        tk.Label(self,**w_preset,text='Extension:').place(relx=.5,rely=.5,anchor='c')

        self.entry=tk.Entry(self,font=font,textvariable=self.search_v)
        self.entry.place(relx=.5,rely=.6,anchor='c')

        tk.Button(self,**w_preset,text='Find',command=self.search).place(relx=.8,rely=.6,anchor='c')

        self.count=tk.Label(self)

    def count_sentences(self):
```

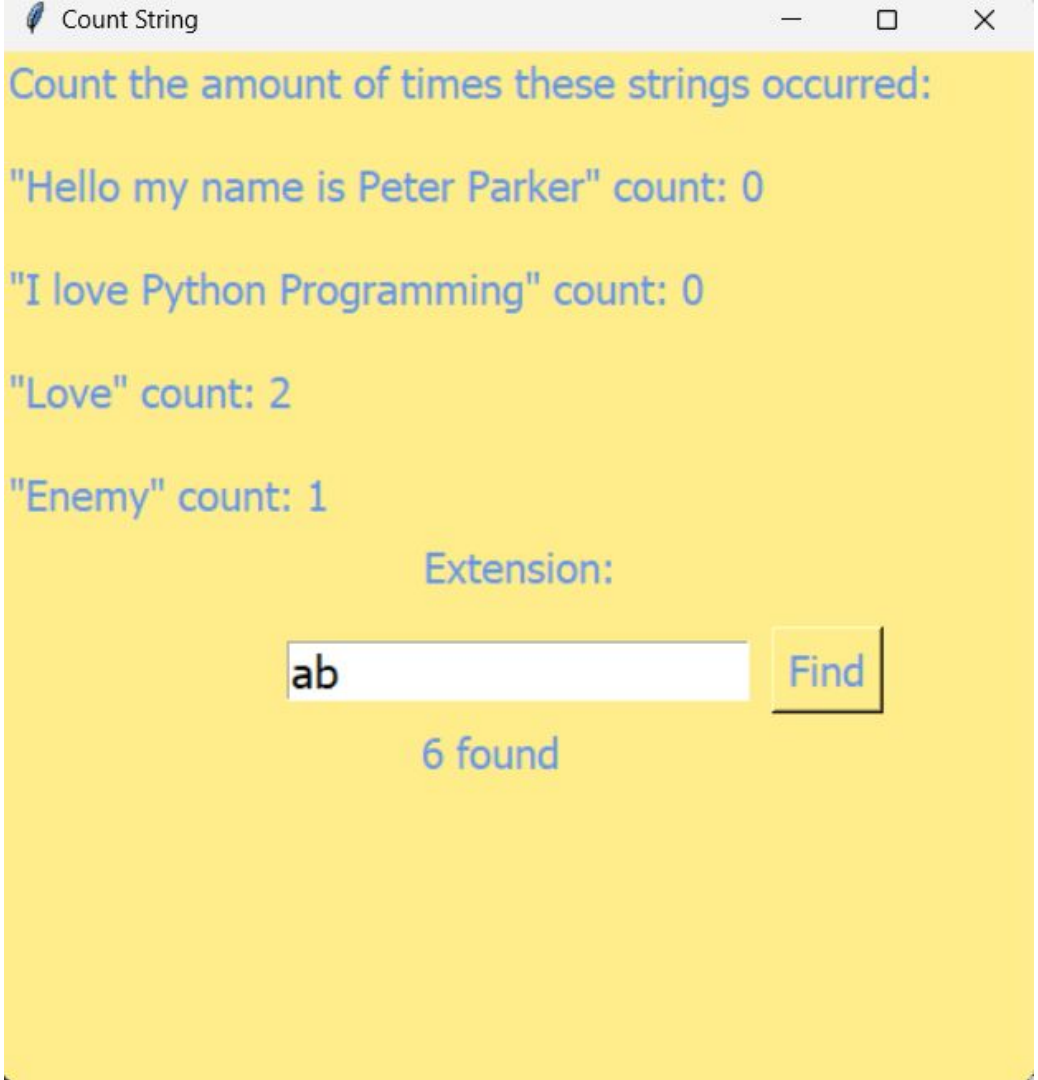
back

Ch4-ex02 code 2

```
    def count_sentences(self):
        file=self.file
        strings_t=self.target_strings

        item_1=self.item_1
        item_2=self.item_2
        item_3=self.item_3
        item_4=self.item_4

        with open(file) as file_handler:
            string_l=file_handler.readlines()
            t_1_c=sum(1 for element in string_l if strings_t[0].lower() in element.lower())
            t_2_c=sum(1 for element in string_l if strings_t[1].lower() in element.lower())
            t_3_c=sum(1 for element in string_l if strings_t[2].lower() in element.lower())
            t_4_c=sum(1 for element in string_l if strings_t[3].lower() in element.lower())

        item_1.configure(text=f'"{strings_t[0]}" count: {t_1_c}')
        item_2.configure(text=f'"{strings_t[1]}" count: {t_2_c}')
        item_3.configure(text=f'"{strings_t[2]}" count: {t_3_c}')
        item_4.configure(text=f'"{strings_t[3]}" count: {t_4_c}')


    def search(self):
        query=self.search_v.get()
        file=self.file


        self.count.destroy()

        with open(file) as file_handler:
            count=sum(1 for element in file_handler if query in element)

            if len(query)==0:
                count=0

            self.count=tk.Label(self,**self.w_preset,text=f'{count} found')
            self.count.place(relx=.4,rely=.65)

if __name__ == '__main__':
    App().mainloop()
```

back

Ch4-ex02 output



back

Ch4-ex03

```
file='Text Files/numbers.txt'

with open(file) as file_handler:
    file_list=file_handler.readlines()

int_file=[int(i) for i in file_list]

print(int_file)
```

```
[58, 19, 4, 46, 52, 33, 36, 96, 15, 97, 37, 42, 88, 83, 62, 91, 79, 25, 1, 10, 3
9, 47, 16, 55, 2, 69, 59, 90, 43, 82, 74, 80, 92, 38, 12, 50, 34, 87, 32, 21, 23
, 78, 9, 89, 71, 68, 93, 57, 86, 5, 8, 81, 70, 18, 20, 67, 3, 56, 11, 48, 99, 98
, 17, 14, 30, 76, 63, 40, 61, 13, 64, 60, 100, 6, 66, 84, 41, 22, 72, 45, 94, 75
, 95, 26, 49, 51, 7, 53, 29, 31, 28, 27, 85, 24, 65, 44, 77, 54, 73, 35]
```

back

Ch4-ex04 code 1

```
import tkinter as tk

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title('Return Letter Count')
        self.geometry('500x500')
        self['bg']='lightgoldenrod1'
        self.entry_1_v=tk.StringVar()
        #vars
        self.font=('Tahoma',16)
        self.w_preset={'font':('Tahoma',15),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.file='Text Files/sentences.txt'

        self.label_1=tk.Label(self,**self.w_preset,text='Find any occurences of the input string in File')
        self.label_1.place(relx=.5,anchor='n')

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.entry_1_v)
        self.entry_1.place(relx=.5,rely=.1,anchor='n')

        self.button_1=tk.Button(self,**self.w_preset,text='Search',command=self.search)
        self.button_1.place(relx=.81,rely=.09,anchor='n')

        self.label_2=tk.Label(self)

        self.label_3=tk.Label(self,**self.w_preset,text='File contents')
        self.label_3.place(relx=.5,rely=.3,anchor='n')

        self.text_1=tk.Text(self,font=self.font)
        self.text_1.place(relx=.5,rely=.4,anchor='n',relwidth=.9,relheight=.55)
        self.insert_content()

        self.scroll=tk.Scrollbar(self,orient='vertical')
        self.scroll.place(relx=.966,rely=.4,anchor='n',relheight=.55)

        self.scroll.config(command=self.text_1.yview)
        self.text_1.config(yscrollcommand=self.scroll.set)


    def search(self):
        query=self.entry_1_v.get()

        file=self.file

        self.label_2.destroy()
```

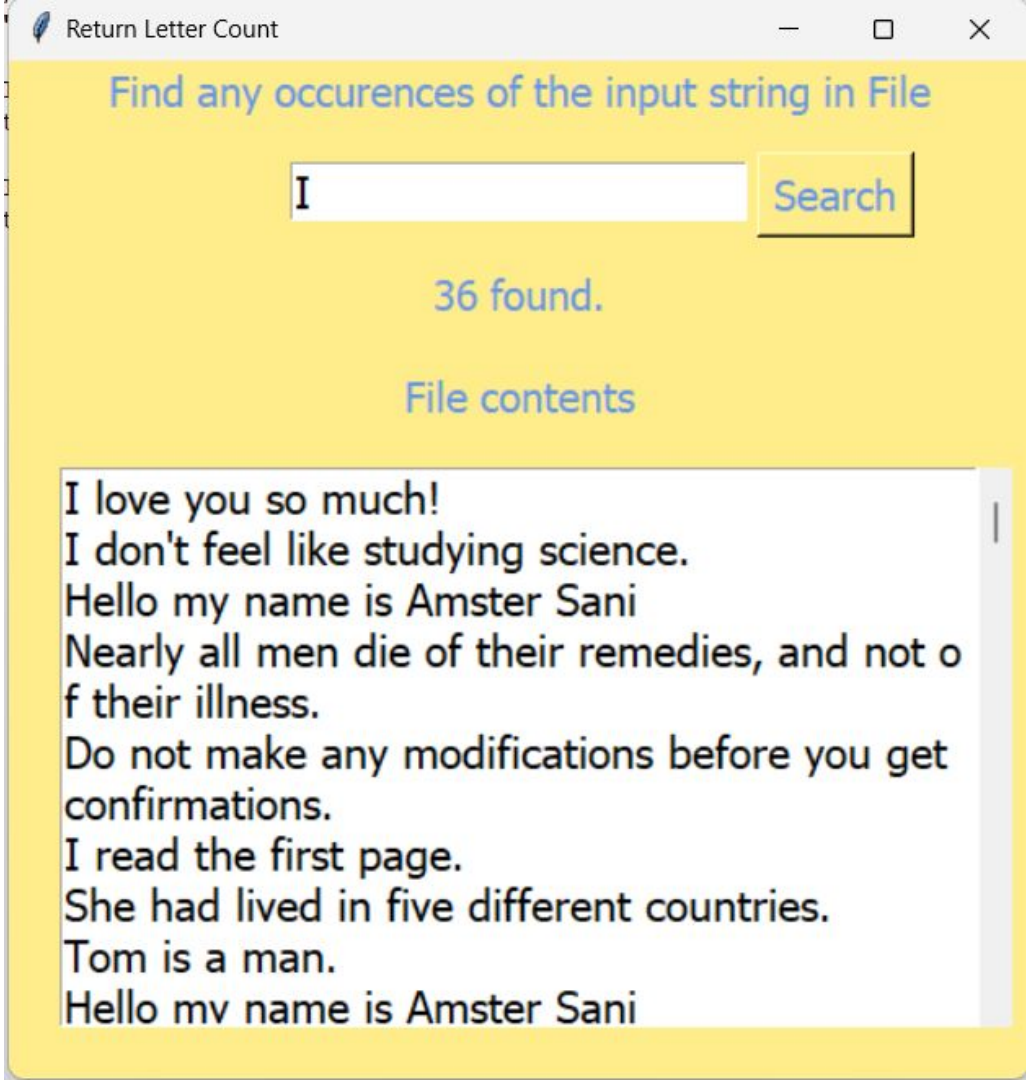
back

Ch4-ex04 code 2

```
    def search(self):
        query=self.entry_1_v.get()

        file=self.file

        self.label_2.destroy()


        with open(file) as file_handler:
            file_list=file_handler.readlines()
        count=sum(1 for element in file_list if query in element)

        if len(query)==0 or ' ' in query:
            count=0

        self.label_2=tk.Label(self,**self.w_preset,text=f'{count} found.')
        self.label_2.place(relx=.5,rely=.2,anchor='n')

    def insert_content(self):
        file=self.file
        text=self.text_1

        with open(file) as file_handler:
            content=file_handler.read()
        text.insert(tk.END,content)
if __name__=='__main__':
    App().mainloop()
```

back

Ch4-ex04 output



back

Ch4-ex05 code 1

```
import re
import tkinter as tk
from tkinter import messagebox

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title('Validate Password')
        self.geometry('500x500')
        self['bg']='lightgoldenrod1'

        #vars
        self.font=('Tahoma',16)
        self.w_preset={'font':('Tahoma',15),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.p_preset={'relx':'.5','anchor':'n'}
        self.password_v=tk.StringVar()

        # default # tries
        self.tries=5
        # state default value
        self.state=True

        self.label_1=tk.Label(self,**self.w_preset,text='Enter Password')
        self.label_1.place(**self.p_preset)

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.password_v)
        self.entry_1.place(**self.p_preset,rely=.1)

        self.button_1=tk.Button(self,font=self.font,bg='red',text='Enter',command=self.enter_pass_brute)
        self.button_1.place(relx=.8,rely=.08,anchor='n')

        self.entry_2=tk.Label(self,**self.w_preset,text='Must follow these rules:')
        self.entry_2.place(**self.p_preset,rely=.2)

        self.label_2=tk.Label(self,**self.w_preset,text='Minimum 1 letter from a-z')
        self.label_2.place(**self.p_preset,rely=.3)

        self.label_3=tk.Label(self,**self.w_preset,text='Minimum 1 digit from 0-9')
        self.label_3.place(**self.p_preset,rely=.4)

        self.label_4=tk.Label(self,**self.w_preset,text='Minimum 1 letter from A-Z')
        self.label_4.place(**self.p_preset,rely=.5)

        self.label_5=tk.Label(self,**self.w_preset,text='1 of these @#$ symbols')
        self.label_5.place(**self.p_preset,rely=.6)

        self.label_6=tk.Label(self,**self.w_preset,text='Min/Max length: 6/12')
```

back

Ch4-ex05 code 2

```
        self.label_6=tk.Label(self,**self.w_preset,text='Min/Max length: 6/12')
        self.label_6.place(**self.p_preset,rely=.7)

    def enter_pass_brute(self):
        password=self.password_v.get()
        self.state=True
        # ord returns the ASCII code of the argument. Which is a number we can iterate through. We then convert
        letters_u=[chr(i) for i in range(ord('A'), ord('Z') +1)]
        letters_l=[x.lower() for x in letters_u]
        numbers=[str(y) for y in range(10)]
        special=['@','#','$']

        letters_u_c=sum(1 for char in password if char in letters_u)
        letters_l_c=sum(1 for char in password if char in letters_l)
        numbers_c=  sum(1 for char in password if char in numbers)
        special_c=  sum(1 for char in password if char in special)
        total_c=[letters_u_c,letters_l_c,numbers_c,special_c]

        # 1st condition of password. check min and max lengths. 2nd condition of password. check if certain cha
        if len(password)<6 or len(password)>12 or 0 in total_c:
                self.state=False

        # 3rd condition. check if password has invalid chars
        for char in password:
            if char not in letters_u and char not in letters_l and char not in numbers and char not in special:
                self.state=False

        self.checkstate()
        self.checktries()

    def enter_pass_regex(self):
        password=self.password_v.get()
        condition=r'(?=.*?[A-Z])(?=.*?\d)(?=.*?[a-z])(?=.*?[@#$])^[A-Za-z\d@#$]{6,12}$'
        self.state=True
        if re.match(condition,password)==None:
            self.state=False

        self.checkstate()
        self.checktries()

    def checkstate(self):
        if self.state==False:
            self.tries-=1
            messagebox.showerror('Fail',f'Incorrect password. Remaining attempts: {self.tries}.')
        else:
            messagebox.showinfo('Success','You successfully broke into a bank!')
```

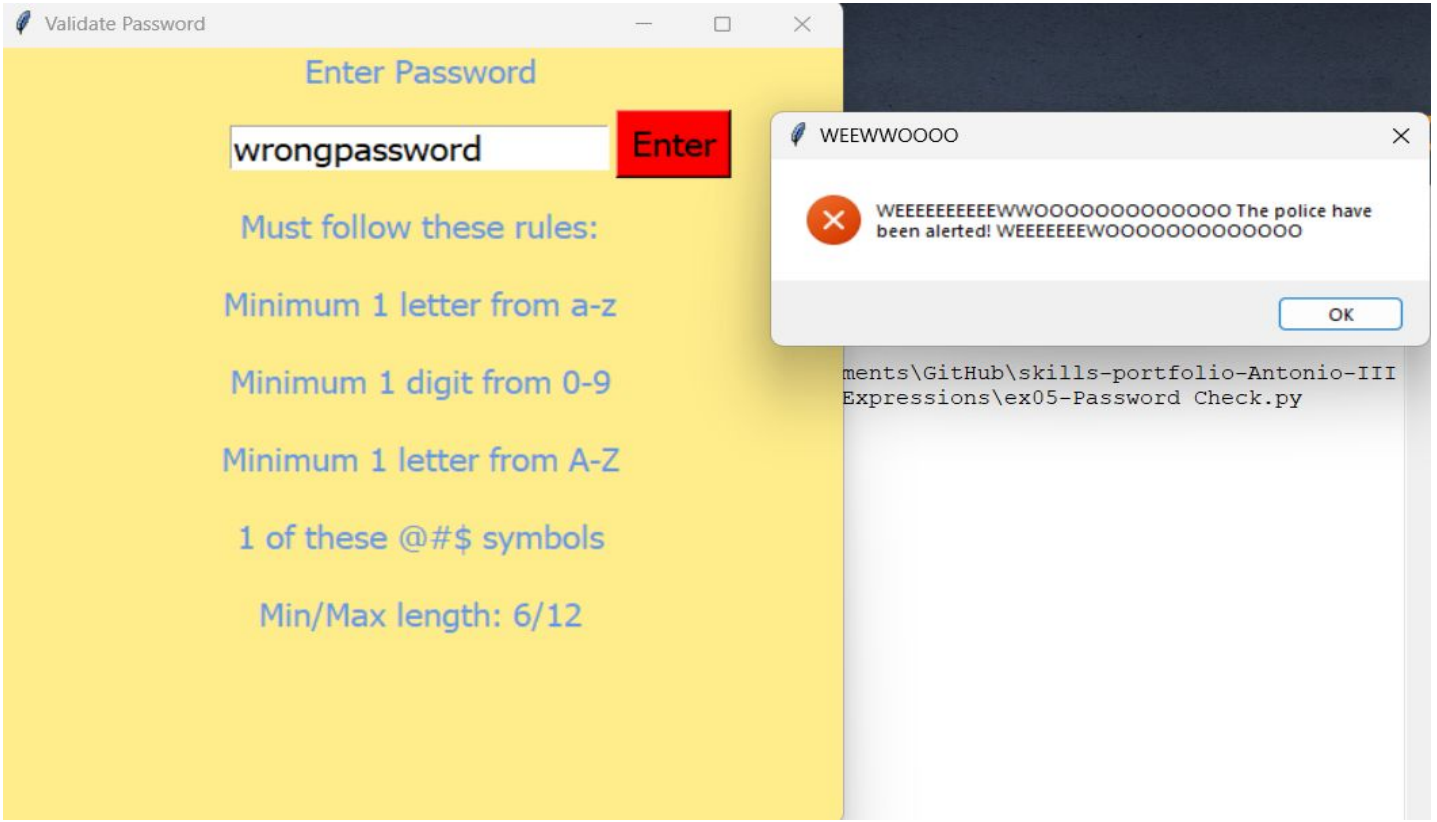
back

Ch4-ex05 code 3

```
            messagebox.showerror('Fail',f'Incorrect password. Remaining attempts: {self.tries}.')
        else:
            messagebox.showinfo('Success','You successfully broke into a bank!')

    def checktries(self):
        if self.tries==0:
            messagebox.showerror('WEEWWOOOO','WEEEEEEEEEEWWOOOOOOOOOOOOO The police have been alerted! WEEEEEEEWOOOOOOOOOOOOO')
            # close app
            self.destroy()

if __name__=='__main__':
    App().mainloop()
```

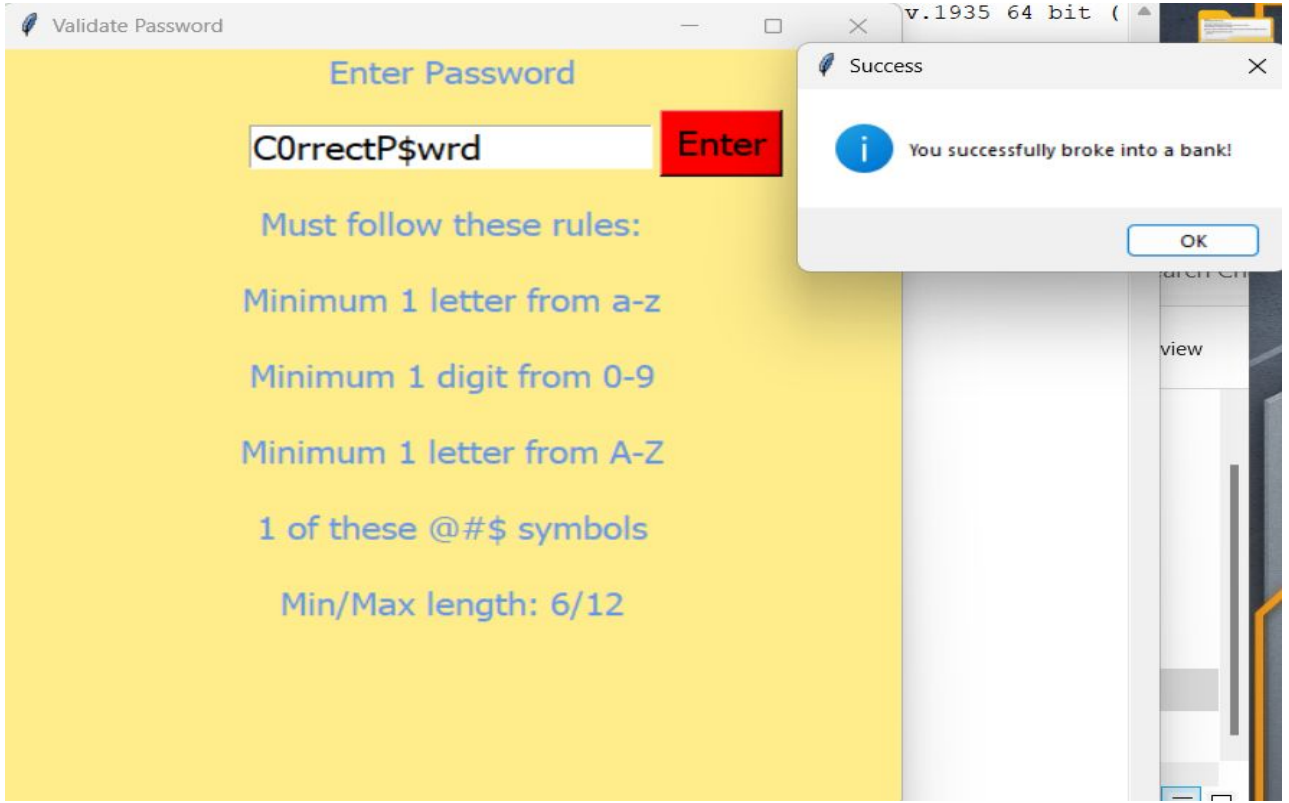
back

Ch4-ex05 output 1



back

Ch4-ex05 output 2



back

Ch5-ex01 code 1

```
import tkinter as tk
from PIL import ImageTk, Image

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title('WOof woof')
        self.geometry('500x500')

        self['bg']='lightgoldenrod1'

        # vars
        self.w_preset={'font':('Tahoma',16),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.p_preset={'relx':.75,'anchor':'n'}

        # dog instances (backend)
        self.dog_1=self.Dog('Victor',1)
        self.dog_2=self.Dog('Donald',42)

         # dog images
        self.dog_1_label=tk.Label(self,image=self.dog_1.image())
        self.dog_1_label.place(x=0)

        self.dog_2_label=tk.Label(self,image=self.dog_2.image())
        self.dog_2_label.place(rely=.5)

        # texts
        self.label_3=tk.Label(self,**self.w_preset,text='Dog Characteristics')
        self.label_3.place(**self.p_preset)


        # details
        self.dog_1_details=tk.Label(self,**self.w_preset,text=self.dog_1.details())
        self.dog_1_details.place(**self.p_preset,rely=.1)
        self.bark_1=tk.Label(self)

        self.dog_2_details=tk.Label(self,**self.w_preset,text=self.dog_2.details())
        self.dog_2_details.place(**self.p_preset,rely=.6)

        self.bark_2=tk.Label(self)


        # button
```

back

Ch5-ex01 code 2

```
        # button
        self.button_1=tk.Button(self,**self.w_preset,text='Bark',command=self.bark)
        self.button_1.place(**self.p_preset,rely=.9)

    class Dog():
        def __init__(self,name,age):
            self.name=name
            self.age=age
            self.dog_img=None
        def details(self):
            self.details=f'Name: {self.name}\nAge: {self.age}'
            return self.details

        def resize(self,directory,width,height):
            b_img= Image.open(directory)
            r_img=b_img.resize((width,height))
            self.dog_img=ImageTk.PhotoImage(r_img)
            return self.dog_img

        def image(self):
            if self.age<10:
                return self.resize('Images/small dog.jpg',250,250)
            else:
                return self.resize('Images/big dog.jpg',250,250)

    def bark(self):
        def bark_1():
            self.bark_1=tk.Label(self,**self.w_preset,text='WOOF WOOF')
            self.bark_1.place(**self.p_preset,rely=.25)

        def bark_2():
            self.bark_2=tk.Label(self,**self.w_preset,text='WOOF WOOF')
            self.bark_2.place(**self.p_preset,rely=.75)


        # destroy existing labels. these 2 have to be created by the end of this funct
        self.bark_1.destroy()
        self.bark_2.destroy()

        if self.dog_1.age>self.dog_2.age:
            bark_1()

            # we need to create label because the top of the function code block requ
            self.bark_2=tk.Label(self)

        elif self.dog_2.age>self.dog_1.age:
            # we need to create label because the top of the function code block requ
```

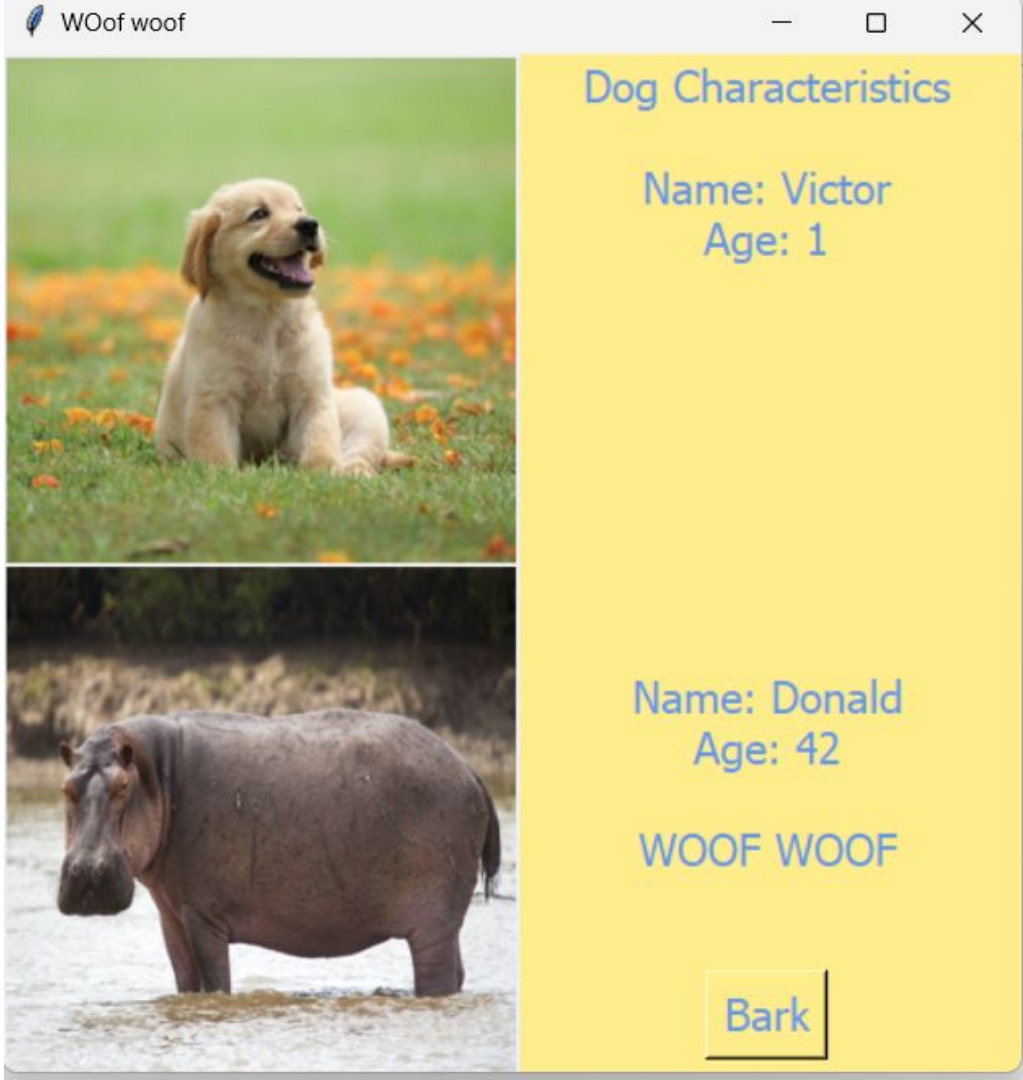
back

Ch5-ex01 code 3

```
        elif self.dog_2.age>self.dog_1.age:
            # we need to create label because the
            self.bark_1=tk.Label(self)

            bark_2()


if __name__=='__main__':
    App().mainloop()
```

back

Ch5-ex01 output



back

Ch5-ex02 code 1

```
import tkinter as tk
from PIL import ImageTk,Image
import re
class Main(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title('Student Class')
        self.geometry('600x600')

        self['bg']='lightgoldenrod1'

        #vars
        self.font=('Tahoma',16)
        self.w_preset={'font':('Tahoma',16),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.p_preset={'relx':.75,'anchor':'n'}
        self.p_preset_2={'relx':.86,'anchor':'n','relwidth':.1,'relheight':.04}
        self.mark_1_input=tk.StringVar()
        self.mark_2_input=tk.StringVar()
        self.mark_3_input=tk.StringVar()

        # objects
        self.student_1=self.Student('You',mark1='69',mark2='80',mark3='75')
        self.student_2=self.Student('Me')

        # widgets student 1 guis (left)
        self.student_1_image=self.resize('Images/cringe(you).png')
        self.student_1_label=tk.Label(self,**self.w_preset,image=self.student_1_image)
        self.student_1_label.place(x=0)

        self.student_2_image=self.resize('Images/based(me).jpg')
        self.student_2_label=tk.Label(self,**self.w_preset,image=self.student_2_image)
        self.student_2_label.place(rely=.5)

        self.student_1_details=tk.Label(self,**self.w_preset,text=self.student_1.display())
        self.student_1_details.place(**self.p_preset)

        #student 2 guis (right)
        self.student_2_details=tk.Label(self,**self.w_preset,text=self.student_2.display(mode=1))
        self.student_2_details.place(**self.p_preset,rely=.5)

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.mark_1_input)
        self.entry_1.place(**self.p_preset_2,rely=.59)

        self.entry_2=tk.Entry(self,font=self.font,textvariable=self.mark_2_input)
        self.entry_2.place(**self.p_preset_2,rely=.67)
```

back

Ch5-ex02 code 2

```
        #student 2 guis (right)
        self.student_2_details=tk.Label(self,**self.w_preset,text=self.student_2.display(mode=1))
        self.student_2_details.place(**self.p_preset,rely=.5)

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.mark_1_input)
        self.entry_1.place(**self.p_preset_2,rely=.59)

        self.entry_2=tk.Entry(self,font=self.font,textvariable=self.mark_2_input)
        self.entry_2.place(**self.p_preset_2,rely=.67)

        self.entry_3=tk.Entry(self,font=self.font,textvariable=self.mark_3_input)
        self.entry_3.place(**self.p_preset_2,rely=.75)



        self.button_1=tk.Button(self,**self.w_preset,text='Get Average',command=self.studentupdate)
        self.button_1.place(**self.p_preset,rely=.9)


    def resize(self,directory):
        b_img=Image.open(directory)
        r_img=b_img.resize((300,300))
        self.student_img=ImageTk.PhotoImage(r_img)
        return self.student_img

    def studentupdate(self):
        self.list=[self.student_2_details,self.entry_1,self.entry_2,self.entry_3]
        # widget destroy
        for i in self.list:
            i.destroy()

        # object
        self.student_2=self.Student('Me',mark1=self.mark_1_input.get(),mark2=self.mark_2_input.get(),mark3=self.mark_3_input.get())

        # widget
        self.student_2_details=tk.Label(self,**self.w_preset,text=self.student_2.display(mode=1))
        self.student_2_details.place(**self.p_preset,rely=.5)

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.mark_1_input)
        self.entry_1.place(**self.p_preset_2,rely=.59)

        self.entry_2=tk.Entry(self,font=self.font,textvariable=self.mark_2_input)
        self.entry_2.place(**self.p_preset_2,rely=.67)

        self.entry_3=tk.Entry(self,font=self.font,textvariable=self.mark_3_input)
        self.entry_3.place(**self.p_preset_2,rely=.75)
```

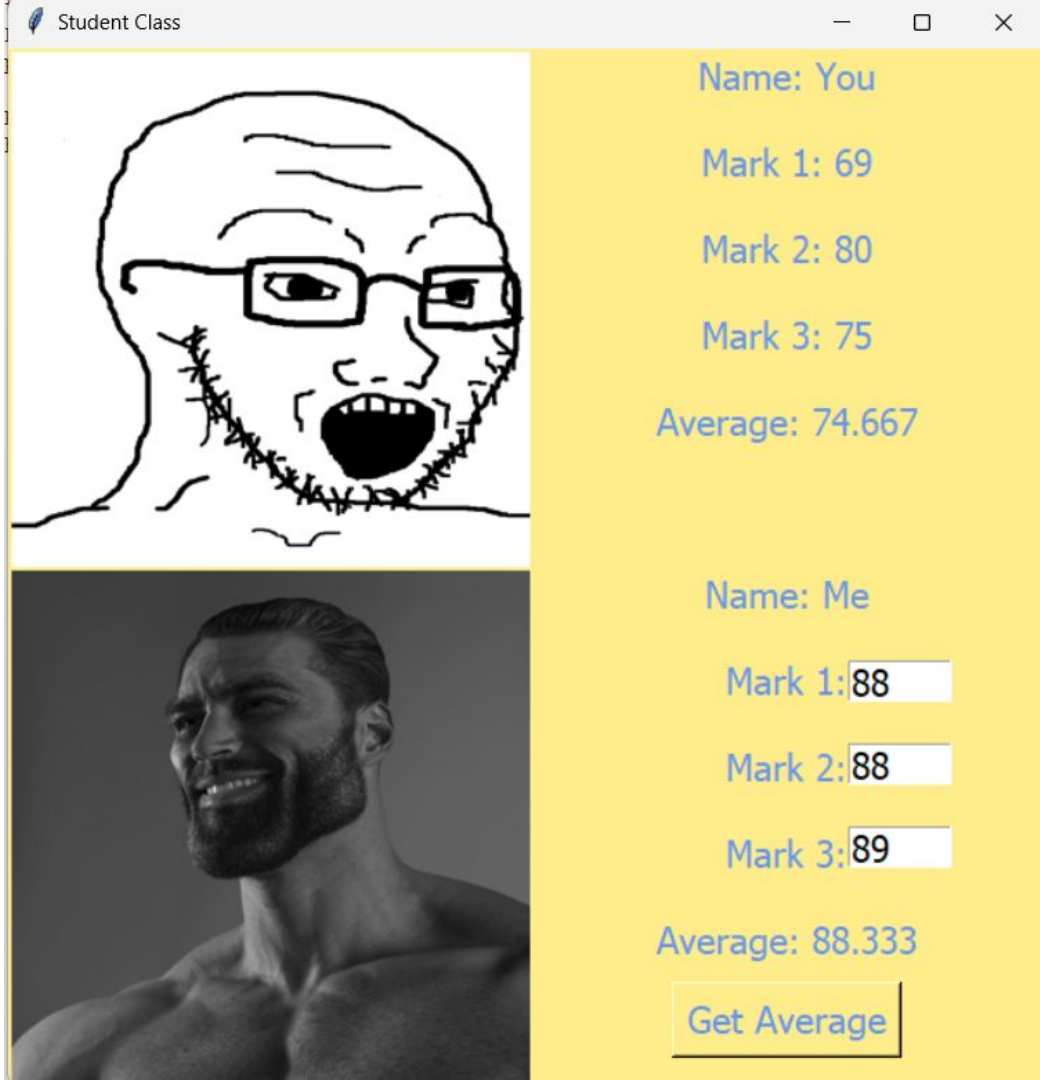
back

Ch5-ex02 code 3

```
        self.entry_3=tk.Entry(self,font=self.font,textvariable=self.mark_3_input)
        self.entry_3.place(**self.p_preset_2,rely=.75)

    class Student():
        def __init__(self,name,**kwargs):
            self.name=name
            # FIX THIS SHIT BRUG
            self.mark1=kwargs.get('mark1','')
            self.mark2=kwargs.get('mark2','')
            self.mark3=kwargs.get('mark3','')

        def calcGrade(self):
            self.list=[self.mark1,self.mark2,self.mark3]

            for i in self.list:
                if not re.match(r'^\d+$',str(i)):
                    self.list[self.list.index(i)]=0


            return round(sum(int(i) for i in self.list)/len(self.list),3)

        def display(self,**kwargs):
            self.operation=kwargs.get('mode','')
            if self.operation!='':
                return f'Name: {self.name}\n\nMark 1:\n\nMark 2:\n\nMark 3:\n\nAverage: {self.calcGrade()}'
            else:
                return f'Name: {self.name}\n\nMark 1: {self.mark1}\n\nMark 2: {self.mark2}\n\nMark 3: {self.mark3}\n\nAverage: {self.calcGrade()}'



f __name__=='__main__':
    Main().mainloop()
```

back

Ch5-ex02 output



back

Ch5-ex03 code 1

```
import tkinter as tk
class App(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title('Employee Class')
        self.geometry('500x500')

        self['bg']='lightgoldenrod1'
        self.font=('Tahoma',16)
        self.w_preset={'font':('Tahoma',16),'bg':'lightgoldenrod1','fg':'cornflowerblue'}
        self.e_p_preset={'rely':.8,'relwidth':.2}

        self.e_ind=tk.IntVar()
        self.e_nam=tk.StringVar()
        self.e_pos=tk.StringVar()
        self.e_sal=tk.StringVar()
        self.e_id=tk.StringVar()

        self.employee_1=self.Employee(name='Antonio',position='CEO',salary=12000,eid=1)
        self.employee_2=self.Employee(name='Miguel',position='Manager',salary=1000,eid=2)
        self.employee_3=self.Employee(name='Jo',position='Senior Dev',salary=2000,eid=3)
        self.employee_4=self.Employee(name='Intal',position='Junior Dev',salary=300,eid=4)
        self.employee_5=self.Employee(name='III',position='Intern',salary=100,eid=5)

        # employee l
        self.employee_list=[self.employee_1,self.employee_2,self.employee_3,self.employee_4,self.employee_5]

        # widget l
        self.widget_list=[]

        self.outputStart()


        # widgets

        self.label_1=tk.Label(self,**self.w_preset,text='Name')
        self.label_1.place(x=0)

        self.label_2=tk.Label(self,**self.w_preset,text='Position')
        self.label_2.place(relx=.25)

        self.label_3=tk.Label(self,**self.w_preset,text='Salary')
        self.label_3.place(relx=.5)

        self.label_4=tk.Label(self,**self.w_preset,text='ID')
        self.label_4.place(relx=.75)
```

back

Ch5-ex03 code 2

```
        self.label_4=tk.Label(self,**self.w_preset,text='ID')
        self.label_4.place(relx=.75)

        self.label_5=tk.Label(self,**self.w_preset,text='Index')
        self.label_5.place(relx=.5,rely=.6,anchor='n')

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.e_ind)
        self.entry_1.place(relx=.5,rely=.7,anchor='n',relwidth=.2)

        self.entry_2=tk.Entry(self,font=self.font,textvariable=self.e_nam)
        self.entry_2.place(**self.e_p_preset)

        self.entry_3=tk.Entry(self,font=self.font,textvariable=self.e_pos)
        self.entry_3.place(**self.e_p_preset,relx=.25)

        self.entry_4=tk.Entry(self,font=self.font,textvariable=self.e_sal)
        self.entry_4.place(**self.e_p_preset,relx=.5)

        self.entry_5=tk.Entry(self,font=self.font,textvariable=self.e_id)
        self.entry_5.place(**self.e_p_preset,relx=.75)

        self.button_1=tk.Button(self,**self.w_preset,text='Set Data',command=self.setData)
        self.button_1.place(relx=.3,rely=.9,anchor='n')

        self.button_2=tk.Button(self,**self.w_preset,text='Print Data',command=self.getData)
        self.button_2.place(relx=.6,rely=.9,anchor='n')


class Employee():
    def __init__(self,name,position,salary,eid):
        self.name=name
        self.position=position
        self.salary=salary
        self.id=eid
def outputStart(self):
    self.incy=.1
    for classes in self.employee_list:
        self.incx=.25

        self.display_name=tk.Label(self,**self.w_preset,text=classes.name)
        self.display_name.place(rely=self.incy)

        self.display_position=tk.Label(self,**self.w_preset,text=classes.position)
        self.display_position.place(relx=self.incx,rely=self.incy)

        self.display_salary=tk.Label(self,**self.w_preset,text=classes.salary)
```

back

Ch5-ex03 code 3

```
        self.display_salary=tk.Label(self,**self.w_preset,text=classes.salary)
        self.display_salary.place(relx=self.incx+.25,rely=self.incy)

        self.display_id=tk.Label(self,**self.w_preset,text=classes.id)
        self.display_id.place(relx=self.incx+.5,rely=self.incy)

        self.incy+=.1

        self.widget_list.append(self.display_name)
        self.widget_list.append(self.display_position)
        self.widget_list.append(self.display_salary)
        self.widget_list.append(self.display_id)

def setData(self):
    self.index=self.e_ind.get()
    self.name=self.e_nam.get()
    self.position=self.e_pos.get()
    self.salary=self.e_sal.get()
    self.id=self.e_id.get()

    if len(self.widget_list)>0:
        for i in self.widget_list:
            i.destroy()

        # remove all elements after destroying widgets
        self.widget_list.clear()

    if self.index in range(len(self.employee_list)):
        self.outputEmployees()

def outputEmployees(self):
    self.employee_list[self.index].name=self.name
    self.employee_list[self.index].position=self.position
    self.employee_list[self.index].salary=self.salary
    self.employee_list[self.index].id=self.id

    self.incy=.1
    for classes in self.employee_list:
        self.checker={}
        for attr_k,attr_v in vars(classes).items():
            if attr_v=='':
                # correction gets added to checker
                self.checker[attr_k]='null'

        # checker updates the current iteration's dictionary
        classes.__dict__.update(self.checker)
```

back

Ch5-ex03 code 4

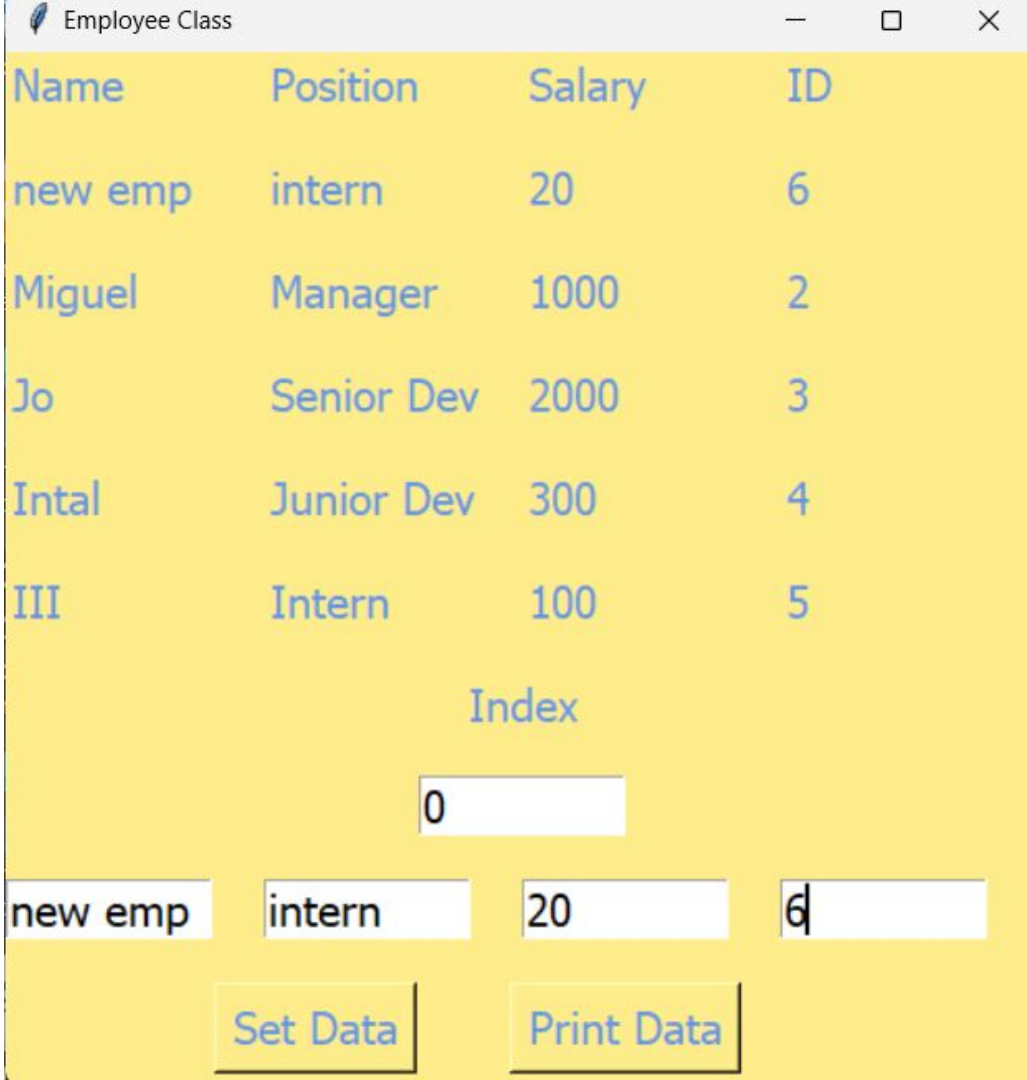
```
        # checker updates the current iteration's dictionary
        classes.__dict__.update(self.checker)


        self.incx=.25

        # output the attributes of the classes post-update
        self.display_name=tk.Label(self,**self.w_preset,text=classes.name)
        self.display_name.place(rely=self.incy)
        self.widget_list.append(self.display_name)

        self.display_position=tk.Label(self,**self.w_preset,text=classes.position)
        self.display_position.place(relx=self.incx,rely=self.incy)
        self.widget_list.append(self.display_position)

        self.display_salary=tk.Label(self,**self.w_preset,text=classes.salary)
        self.display_salary.place(relx=self.incx+.25,rely=self.incy)
        self.widget_list.append(self.display_salary)

        self.display_id=tk.Label(self,**self.w_preset,text=classes.id)
        self.display_id.place(relx=self.incx+.5,rely=self.incy)
        self.widget_list.append(self.display_id)

        self.incy+=.1


    def getData(self):
        print('Employee List:\n')
        for classes in self.employee_list:
            print(f'{classes.name}\n{classes.position}\n{classes.salary}\n{classes.id}\n')




if __name__=='__main__':
    App().mainloop()
```

back

Ch5-ex03 output 1

back

Ch5-ex03 output 2



back

Ch5-ex03 output 3

```
Employee List:

new emp
intern
20
6

Miguel
Manager
1000
2

Jo
Senior Dev
2000
3

Intal
Junior Dev
300
4

III
Intern
100
5
```

back

Ch5-ex04 code 1

```
from PIL import ImageTk,Image
import math
import re
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title='Shapes'
        self.geometry('300x300')

        self['bg']='lightgoldenrod1'

        self.font=('Tahoma',12)
        self.w_preset={'font':('Tahoma',12),'bg':'lightgoldenrod1','fg':'cornflowerblue'

        self.images=[]

        self.param_1=tk.IntVar()
        self.param_2=tk.IntVar()

        self.shapes=['circle','triangle','rectangle']

        # widgets

        self.place_shapes()

        # label
        self.place_widgets()


    def resize_img(self,directory):
        b_img=Image.open(directory)
        r_img=b_img.resize((100,100))
        # always keep the photoimage in a persisting variable ie doesn't get garbage col
        self.img=ImageTk.PhotoImage(r_img)
        # append already-persisting object inside a list so there is always a pointer to
        # This is useful because if this function runs again, it'll replace the attribut
        self.images.append(self.img)

    def place_shapes(self):
        self.incx=0
        for i in self.shapes:
            self.resize_img(f'Images/{i}.png')

            self.place_shape=tk.Label(self,image=self.images[-1])
            self.place_shape.place(relx=self.incx)

            self.incx+=.33
```

back

Ch5-ex04 code 2

```
            self.place_shape=tk.Label(self,image=self.images[-1])
            self.place_shape.place(relx=self.incx)

            self.incx+=.33

    def place_widgets(self):
        self.label_1=tk.Label(self,**self.w_preset,text='radius/base/width')
        self.label_1.place(relx=.3,rely=.7,anchor='s')

        self.label_1=tk.Label(self,**self.w_preset,text='height')
        self.label_1.place(relx=.4,rely=.8,anchor='s')

        self.entry_1=tk.Entry(self,font=self.font,textvariable=self.param_1)
        self.entry_1.place(relx=.7,rely=.7,anchor='s',relwidth=.3)

        self.entry_2=tk.Entry(self,font=self.font,textvariable=self.param_2)
        self.entry_2.place(relx=.7,rely=.8,anchor='s',relwidth=.3)

        self.entry_3=tk.Label(self,**self.w_preset,text='a')
        self.entry_3.place(relx=.165,rely=.4,anchor='c')

        self.entry_4=tk.Label(self,**self.w_preset,text='b')
        self.entry_4.place(relx=.165+.33,rely=.4,anchor='c')

        self.entry_5=tk.Label(self,**self.w_preset,text='c')
        self.entry_5.place(relx=.165+.66,rely=.4,anchor='c')

        self.button_1=tk.Button(self,**self.w_preset,text='Get Area',command=self.get_area)
        self.button_1.place(relx=.5,rely=.95,anchor='s')

    def get_area(self):

        self.input_1=str(self.param_1.get())
        self.input_2=str(self.param_2.get())



        self.input_params=self.input_1+self.input_2


        if re.match(r'^\d+$',self.input_params):

            self.shapes_instance=self.Shapes(param1=self.input_1,param2=self.input_2
                                             )
            self.circle_area=self.Circle(param1=self.shapes_instance.param1,param2=self.input_2).area()
            self.rectangle_area=self.Rectangle(param1=self.shapes_instance.param1,param2=self.input_2).area()
            self.triangle_area=self.Triangle(param1=self.shapes_instance.param1,param2=self.input_2).area()
```

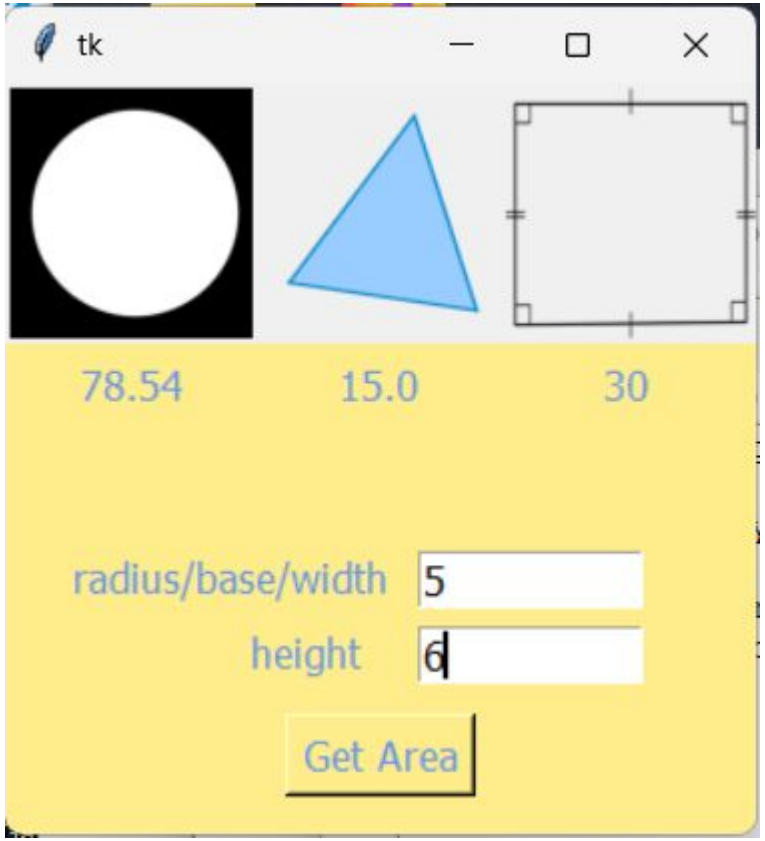
back

Ch5-ex04 code 3

```
        if re.match(r'^\d+$',self.input_params):

            self.shapes_instance=self.Shapes(param1=self.input_1,param2=self.input_2
                                             )
            self.circle_area=self.Circle(param1=self.shapes_instance.param1,param2=self.input_2).area()
            self.rectangle_area=self.Rectangle(param1=self.shapes_instance.param1,param2=self.input_2).area()
            self.triangle_area=self.Triangle(param1=self.shapes_instance.param1,param2=self.input_2).area()


            self.update_label()

    def update_label(self):
        self.entry_3.configure(text=self.circle_area)
        self.entry_4.configure(text=self.triangle_area)
        self.entry_5.configure(text=self.rectangle_area)


    # classes
    class Shapes:
        def __init__(self,**kwargs):
            self.param1=int(kwargs.get('param1'))
            self.param2=int(kwargs.get('param2'))

    class Circle(Shapes):
        def __init__(self,**kwargs):
            super().__init__(**kwargs)

        def area(self):
            return round(math.pi*self.param1**2,3)

    class Triangle(Shapes):
        def __init__(self,**kwargs):
            super().__init__(**kwargs)

        def area(self):
            return (self.param1*self.param2)/2

    class Rectangle(Shapes):
        def __init__(self,**kwargs):
            super().__init__(**kwargs)

        def area(self):
            return self.param1*self.param2

if __name__=='__main__':
    App().mainloop()
```

back

Ch5-ex04 output



back

Ch5-ex05 code 1

```
import tkinter as tk

class App(tk.Tk):
    class Animal:
        def __init__(self,c_type,c_name,c_color,c_age,c_weight,c_noise):
            self.type=c_type
            self.name=c_name
            self.color=c_color
            self.age=c_age
            self.weight=c_weight
            self.noise=c_noise
        def sayHello(self):
            print(f'Hi! My name is {self.name}!')
        def makeNoise(self):
            print(f'This animal says: {self.noise*3}')
        def animalDetails(self):
            print(f'Details',f'Type: {self.type}, Color: {self.color}, Age: {self.age}, Weight: {self.weight}, Noise: {self.noise}')

    def __init__(self):
        super().__init__()
        self.title('Playing around with Classes')
        self.geometry('400x400')

        self['bg']='lightgoldenrod1'

        self.w_preset={'font':('Tahoma',12),'bg':'lightgoldenrod1','fg':'cornflowerblue'}

        self.dog=self.Animal('Dog','Anthony','Blue',12,'10kg','bark')
        self.cat=self.Animal('Cat','Fantano','White',10,'5kg','meow')

        self.animals_list=[self.dog,self.cat]


        self.place_widgets()


    def place_widgets(self):
        self.relx=0

        for i,animal in enumerate(self.animals_list,start=1):
            self.label_widget=tk.Label(self,**self.w_preset,text=f'Animal {i}')
            self.label_widget.place(relx=self.relx)

            self.button_widget=tk.Button(self,**self.w_preset,text='sayHello',command=animal.sayHello)
            self.button_widget.place(relx=self.relx,rely=.1)

            self.button_widget_two=tk.Button(self,**self.w_preset,text='makeNoise',command=animal.makeNoise)
```

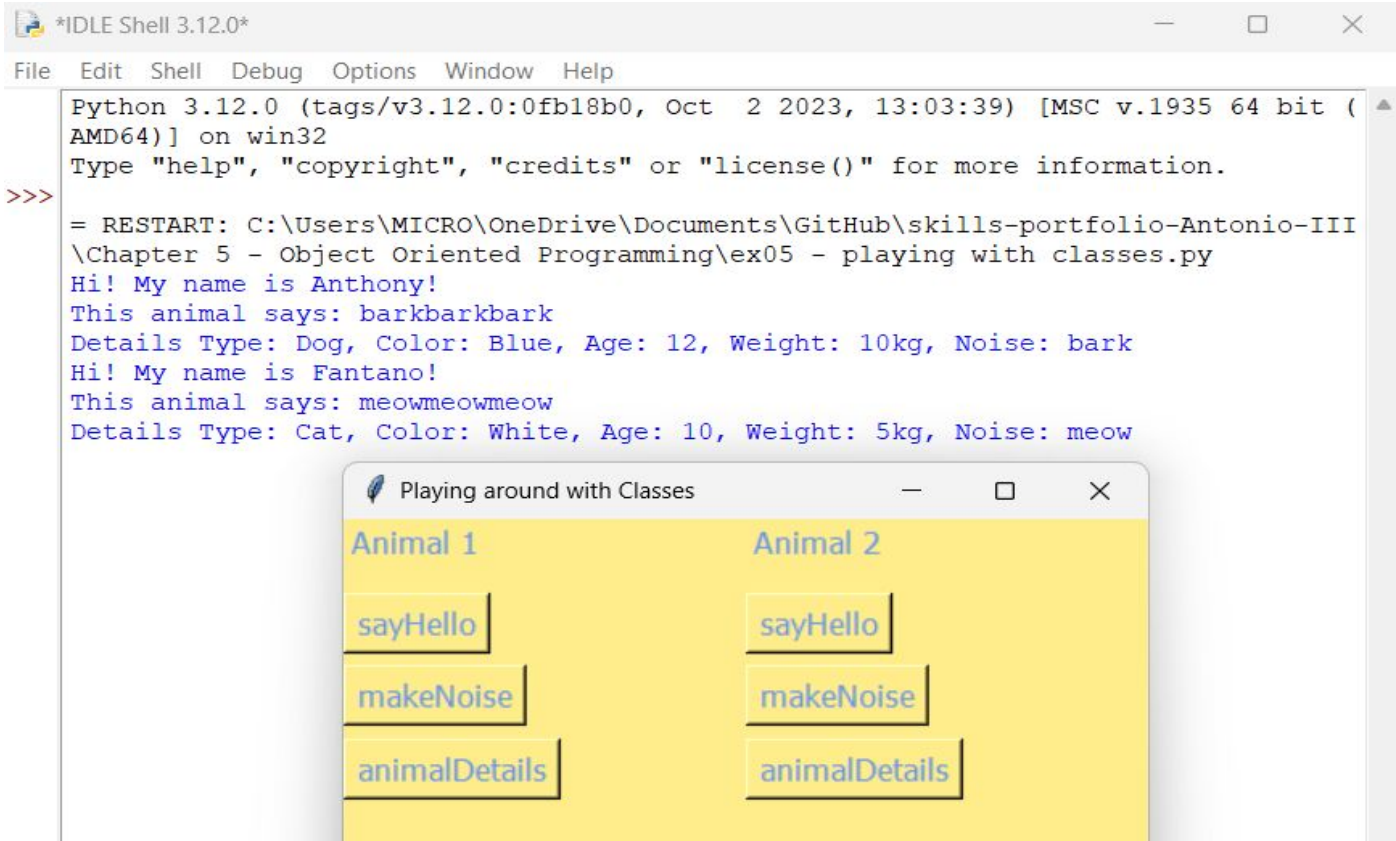
back

Ch5-ex05 code 2

```
        self.button_widget=tk.Button(self,**self.w_preset,text='sayHello',command=animal.sayHello)
        self.button_widget.place(relx=self.relx,rely=.1)

        self.button_widget_two=tk.Button(self,**self.w_preset,text='makeNoise',command=animal.makeNoise)
        self.button_widget_two.place(relx=self.relx,rely=.2)

        self.button_widget_three=tk.Button(self,**self.w_preset,text='animalDetails',command=animal.animalDetails)
        self.button_widget_three.place(relx=self.relx,rely=.3)

        self.relx+=.5



if __name__=='__main__':
    App().mainloop()
```

back

Ch5-ex05 output



back

Ch5-ex06 code 1

```
import tkinter as tk
from tkinter import ttk
import re
class App(tk.Tk):
    class ArithmeticOperations:
        def __init__(self,parent,**kwargs):
            self.param1=kwargs.get('param1')
            self.param2=kwargs.get('param2')

            self.parent=parent

            if re.match(r'^\d+$',self.param1+self.param2):
                self.Calculate()
        def Calculate(self):
            self.mode=self.parent.selected_opt.get()
            self.operation_signs=('+','-','*','/')

            self.selected_operation=self.parent.operations.index(self.mode)

            result=eval(self.param1 +self.operation_signs[self.selected_operation]+ self.param2)

            self.parent.label_w2.configure(text=result)

    def __init__(self):
        super().__init__()
        self.title('Arithmetic Operation')
        self.geometry('400x400')

        self['bg']='lightgoldenrod1'

        self.font=('Tahoma',12)
        self.w_preset={'font':('Tahoma',12),'bg':'lightgoldenrod1','fg':'cornflowerblue'}

        #widgets
        self.label_w=tk.Label(self,**self.w_preset,text='Select Operation')
        self.label_w.place(relx=.5,rely=.1,anchor='c')

        self.selected_opt=tk.StringVar()
        self.operations=('Addition','Subtraction','Multiplication','Division')
        self.options_w=ttk.Combobox(self,font=self.font,values=self.operations,textvariable=self.selected_opt)
        self.options_w.place(relx=.5,rely=.2,anchor='c')

        self.param_one=tk.StringVar()
        self.param_two=tk.StringVar()

        self.entry_w_one=tk.Entry(self,font=self.font,textvariable=self.param_one)
        self.entry_w_one.place(relx=.2,rely=.3,anchor='c',relwidth=.25)
```

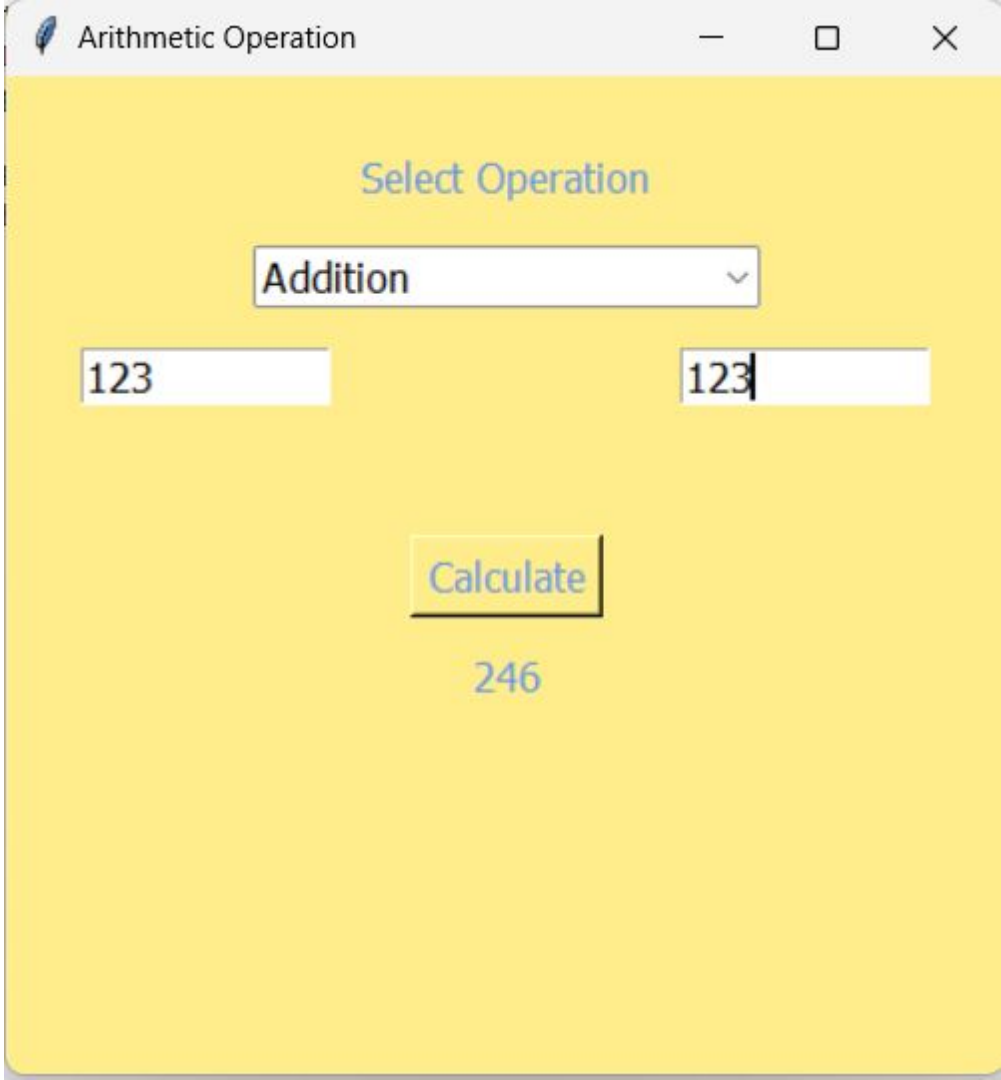
back

Ch5-ex06 code 2

```
        self.entry_w_one=tk.Entry(self,font=self.font,textvariable=self.param_one)
        self.entry_w_one.place(relx=.2,rely=.3,anchor='c',relwidth=.25)

        self.entry_w_two=tk.Entry(self,font=self.font,textvariable=self.param_two)
        self.entry_w_two.place(relx=.8,rely=.3,anchor='c',relwidth=.25)

        self.button_w=tk.Button(self,**self.w_preset,command=self.operation,text='Calculate')
        self.button_w.place(relx=.5,rely=.5,anchor='c')

        self.label_w2=tk.Label(self,**self.w_preset,text='Result will be displayed here')
        self.label_w2.place(relx=.5,rely=.6,anchor='c')

    def operation(self):
        self.ArithmeticOperations(parent=self,param1=self.param_one.get(),param2=self.param_two.get())

if __name__ =='__main__':
    App().mainloop()
```

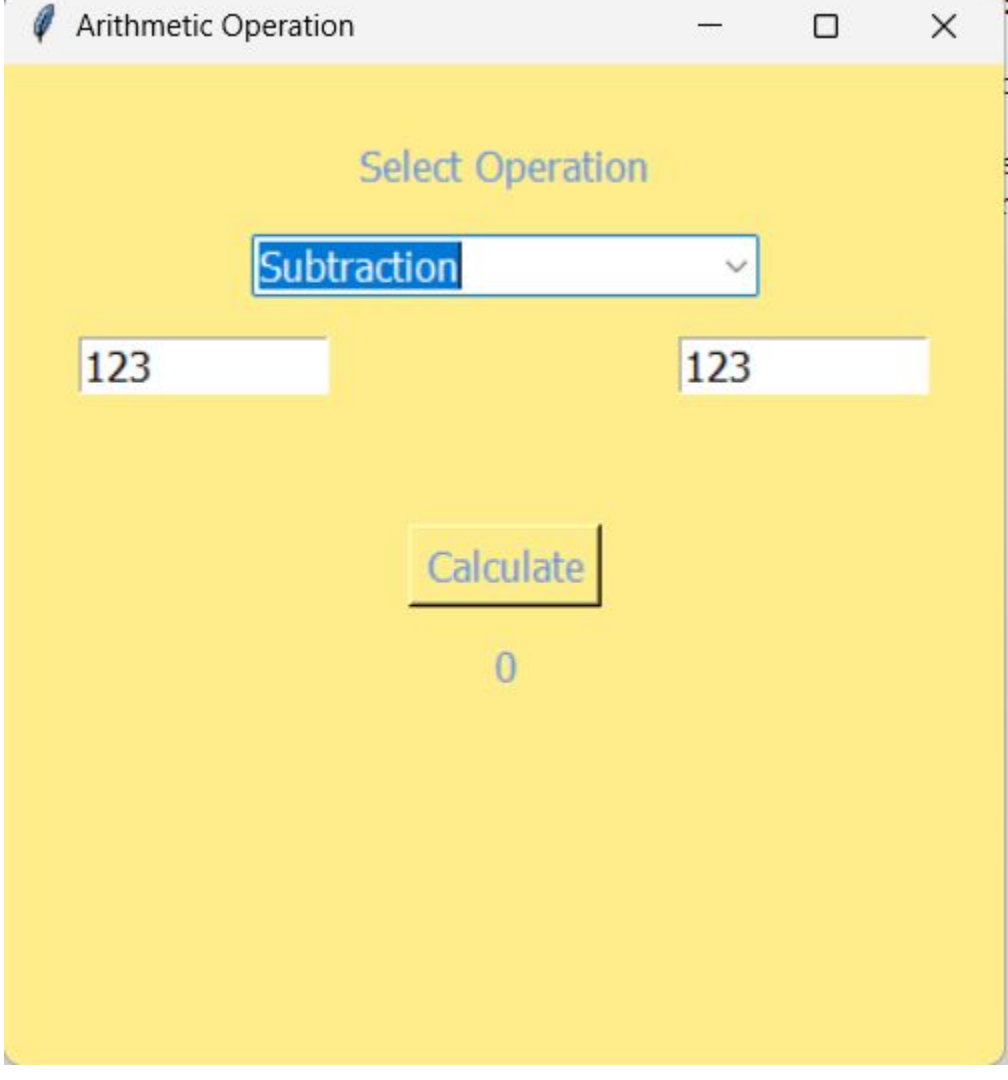
back

Ch5-ex06 output 1



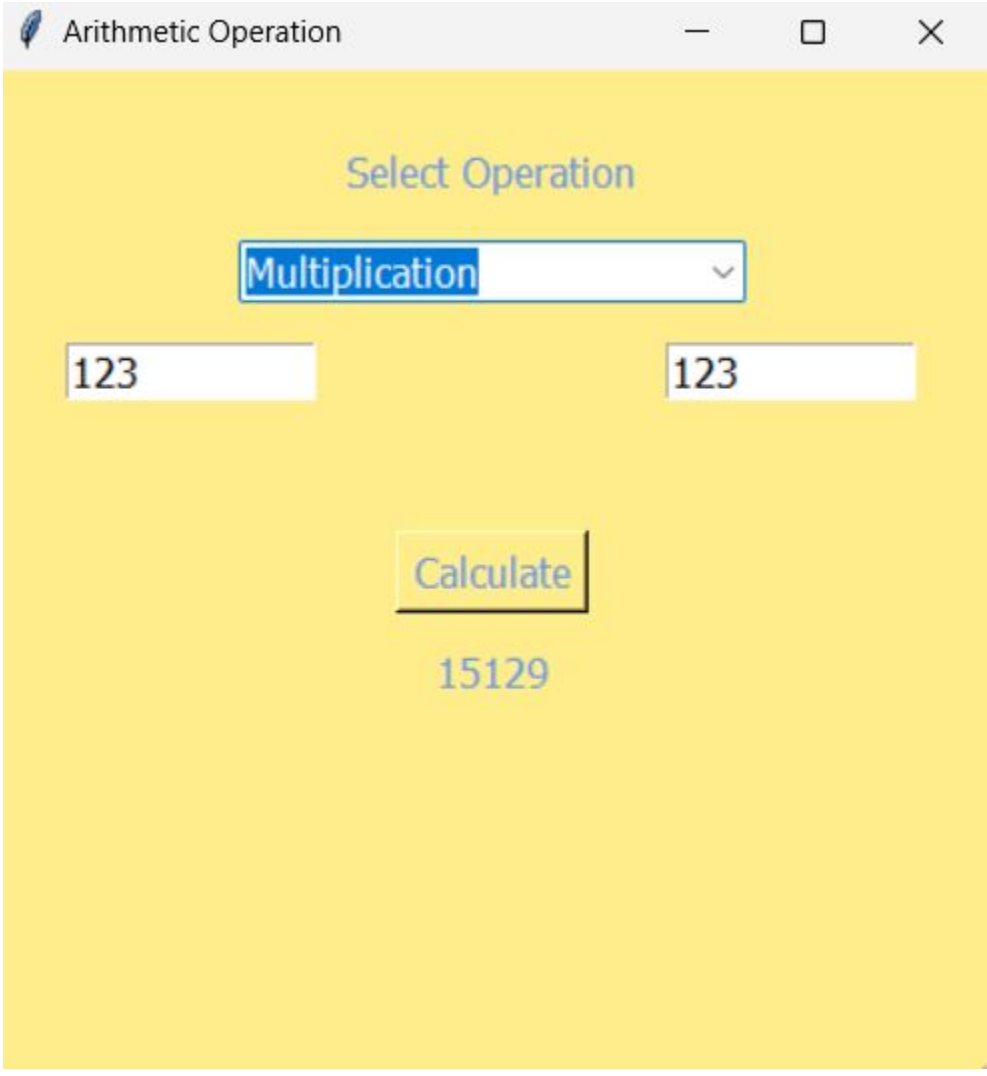
back

Ch5-ex06 output 2



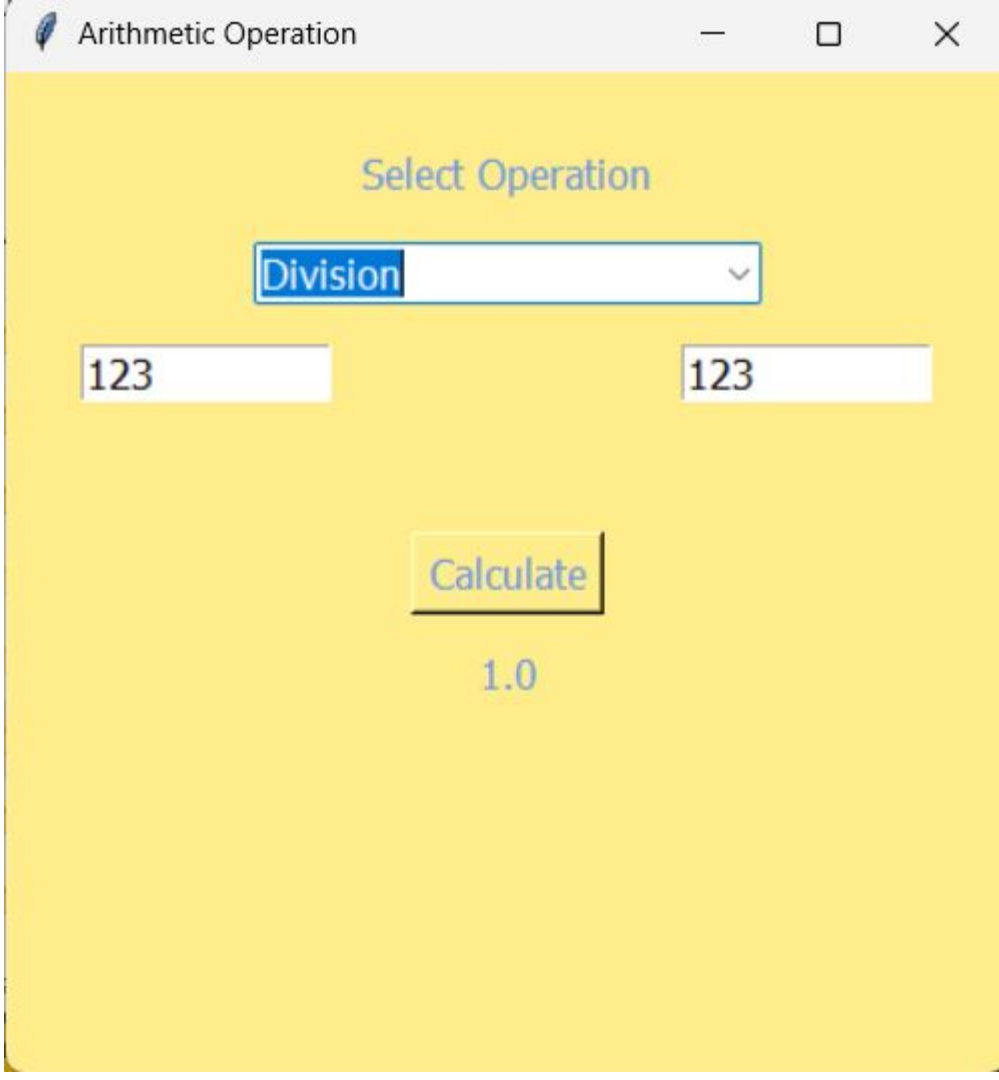
back

Ch5-ex06 output 3



back

Ch5-ex06 output 4



back

Ch6-ex01 code 1

```
import math

class App:
    def __init__(self,**kwargs):
        # round up if non-zero decimal
        self.ceil=kwargs.get('ceil')
        # round down to the nearest whole number. 2.99 rounds to 2
        self.floor=kwargs.get('floor')
        self.factorial=kwargs.get('factorial')
        self.pow=kwargs.get('base')

        self.sqrt=kwargs.get('sqrt')

        self.operations_names=['ceil','floor','factorial','pow','sqrt']
        self.operations_result=[math.ceil(self.ceil),math.floor(self.floor),math.factorial(self.factorial),math.pow(self.pow,3),math.sqrt(self.sqrt)]

    def output(self):
        for operation,result in zip(self.operations_names,self.operations_result):
            print(f'The {operation} of {getattr(self,operation)} is {result}')
            if operation=='pow':
                print(f'{getattr(self,operation)} raised to 3 is {result}')

if __name__=='__main__':
    App(ceil=2.3,floor=2.3,factorial=5,base=2,sqrt=16).output()
```

back

Ch6-ex01 output

```
The ceil of 2.3 is 3
The floor of 2.3 is 2
The factorial of 5 is 120
The pow of 2 is 8.0
2 raised to 3 is 8.0
The sqrt of 16 is 4.0
```

back

Ch6-ex02 code

```
import numpy as np

a = np.array([20,23,82,40,32,15,67,52])
indices_even=np.where(a%2==0)

print('Even numbers in indices: ',indices_even)

# asc sort
print('Array sorted ascending order: ',np.sort(a))

# slice from i 3 to end
print('Sliced from index 3',a[3:])

# slice from i 0 to 4 (inclusive?)
print('Sliced until index 4 inclusive: ',a[:4+1])

# output 32 15 67 using negative nums
print('Negative slicing: ',a[-3:-1])
```

back

Ch6-ex02 output

```
Even numbers in indices:  (array([0, 2, 3, 4, 7], dtype=int64),)
Array sorted ascending order:  [15 20 23 32 40 52 67 82]
Sliced from index 3 [40 32 15 67 52]
Sliced until index 4 inclusive:  [20 23 82 40 32]
Negative slicing:  [15 67]
```

back

Ch6-ex03 code

```
import operator

class App:
    def __init__(self):
        self.quit=False
        self.operations_name=('Addition','Subraction','Multiplication','Division','Modulus Division','Greater Number')
        self.operations_name_len=len(self.operations_name)

        self.foreverloop()

    def foreverloop(self):

        while self.quit!=True:
            self.operation_end=False
            self.user_input=int(input('Enter a number 1-6:\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Modulus\n6. Check greater number\n'))

            if self.user_input in range(1,self.operations_name_len+1): #inclusive
                self.operation(self.user_input)
            else:
                print('Wrong input\n')

            if self.operation_end!=False:
                self.q=input('Would you like to quit? Press q if so\n')

                if self.q.lower()=='q':
                    self.quit=True

    def operation(self,chosen_num):
        self.selected_operation=chosen_num-1 # offset so the index can fit into the list (which has 5 elements, whereas the user can enter the number 6)


        self.num_one=int(input('Enter first num: '))
        self.num_two=int(input('Enter second num: '))

        self.operations=(operator.add(self.num_one,self.num_two),operator.sub(self.num_one,self.num_two),operator.mul(self.num_one,self.num_two),operator.truediv(self.num_one,self.num_two),
                         operator.mod(self.num_one,self.num_two),operator.gt(self.num_one,self.num_two))

        if chosen_num==6:
            print(f'The first number {self.num_one} being greater than {self.num_two} is: {self.operations[self.selected_operation]}\n')
        else:
            print(f'The {self.operations_name[self.selected_operation]} between {self.num_one} and {self.num_two} is {self.operations[self.selected_operation]}\n')

        self.operation_end=True

if __name__=='__main__':
    App()
```

back

Ch6-ex03 output

```
Enter a number 1-6:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
1
Enter first num: 5
Enter second num: 5
The Addition between 5 and 5 is 10

Would you like to quit? Press q if so
t
Enter a number 1-6:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
66
Wrong input

Enter a number 1-6:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Modulus
6. Check greater number
6
Enter first num: 5
Enter second num: 3
The first number 5 being greater than 3 is: True

Would you like to quit? Press q if so
q
```

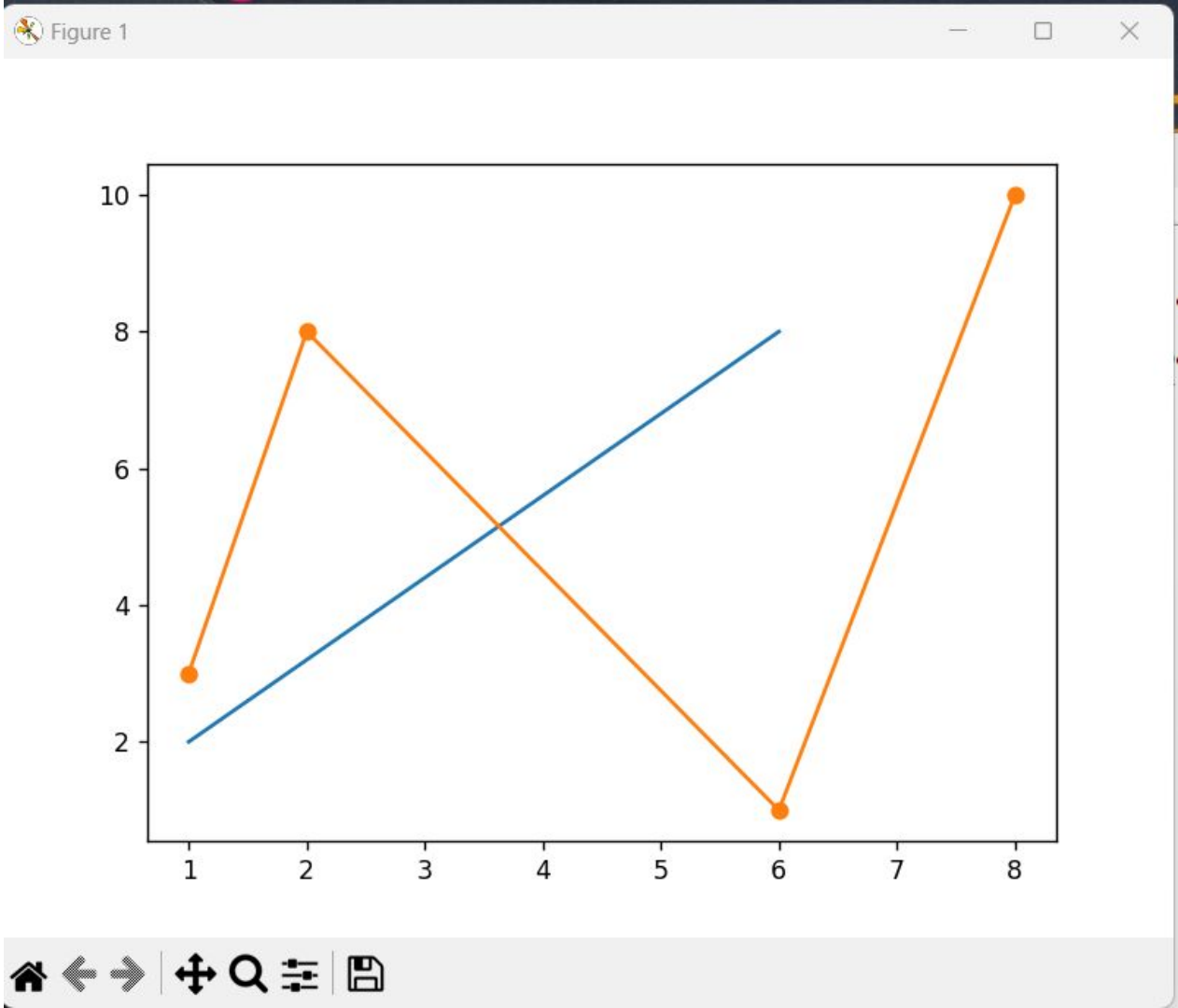
back

Ch6-ex04 code

```
import matplotlib.pyplot as plt

target_one=[(1,2),(6,8)]
x_one=[tuple_elem for list_elem in target_one for tuple_elem in list_elem if list_elem.index(tuple_elem)==0]
y_one=[tuple_elem for list_elem in target_one for tuple_elem in list_elem if list_elem.index(tuple_elem)==1]
plt.plot(x_one,y_one)

target_two=[(1, 3) ,(2, 8), (6, 1) , (8, 10)]
x_two=[tuple_elem for list_elem in target_two for tuple_elem in list_elem if list_elem.index(tuple_elem)==0]
y_two=[tuple_elem for list_elem in target_two for tuple_elem in list_elem if list_elem.index(tuple_elem)==1]
plt.plot(x_two,y_two,marker='o')

plt.show()
```

back

Ch6-ex04 output



back

Ch6-ex05 code

```
import json

class App:
    def __init__(self):
        self.name=input('Enter name: ')
        self.id=input('Enter ID: ')
        self.course=input('Enter Course: ')

        self.dict={"Details of the Student are":{"Name":self.name.title(),"ID":self.id,"Course":self.course}}

        self.output_one()
        self.output_two()

    def write_to_file(self,directory,content):
        with open(directory,'w') as file_handler:
            json.dump(content,file_handler,indent=4)

    def output_dict(self):
        for key in self.dict.keys():
            print(f'\n{key}')

        for key,value in self.dict['Details of the Student are'].items():
            print(f'\t{key}: {value}')

    def output_one(self):
        # you can create a new variable and have it point to the .load() return value of json lib, but I'm ski
        self.write_to_file('Text Files/StudentJson.json',self.dict)

        # output file
        self.output_dict()


    def output_two(self):
        # introduce new dict
        self.course_details={"Course Details":{"Group":"A","Year":2}}
        # update student dictionary with new dict values
        self.dict["Details of the Student are"].update(self.course_details["Course Details"])
        # update file
        self.write_to_file('Text Files/StudentJson.json',self.dict)

        # output file
        self.output_dict()

if __name__=='__main__':
    App()
```

back

Ch6-ex05 output

```
Enter name: antonio
Enter ID: 42
Enter Course: CS

Details of the Student are
        Name: Antonio
        ID: 42
        Course: CS

Details of the Student are
        Name: Antonio
        ID: 42
        Course: CS
        Group: A
        Year: 2
```