**Spencer Olson** **January 24, 2025**
**Lab for Chapter 2 – from Kaefer pp. 36-37**

2.1 Write Python code that uses the input built-in function to ask the user to enter a whole number between 1 and 100. The input function always returns a string value, so use the int built-in function to convert the value entered to an integer data type and square the number that the user entered using the exponentiation operator. Print a message to the user stating the value that they entered and the square of the value that they entered.
**The input function always returns a string.**
{} input your named variable within as a placeholder.

A. Code here:

```
B. user_input = int(input("Enter number 1-100:"))
C. print("The square of " + str(user_input) + " is " +  str(user_input**2))
D.
```

b. If you use AI to write code for this, what must you be careful of (go beyond "checking the code is correct" discussing comments, style, context, etc.)?

c. List and explain all the expressions in the code (include discussion of variables and operators):
**Input is the base expression, where Int is the external expression. The variable (user_input) is a naming expression for a variable.**
**Print is an expression that lists the following string withing its parenthesis. Inside the parenthesis there is concatenated with word outputs along with str() of the named variables.**

d. Explain why the input must be converted to an integer data type:
**To ensure that it is calculated as a number if 2 or more digits, instead of combined in a linear line. You can't square a string.**

e. List example "invalid inputs" and explain how the program would respond to them:
**Words could generate a value error.**
**Decimals would generate a value error as it's not an integer.**

f. Show one alternative way to print the result message and explain the pros and cons of this approach (hint: there are at least two very different ways):

```
# Ask user to enter a qhole number between 1 and 100
GPT_input = ("Enter a number 1-100")

#Convert input to integer and square the number
```

```
GPT_number = int(GPT_input)
GPT_squared_number = GPT_number ** 2

# Print results
print(f"The square of {GPT_number} is {GPT_squared_number}")
```

**Simplifies and labels everything in its own sections.**

extra credit: Explain if a function is an expression:
**No, regular functions are objects or definitions. But calling a function that are linked to numerical variables that result in a return can result in an expression.**
**Lambda however is because it evaluates to a function object.**

2.2 Write Python code that uses the input built-in function to ask the user to enter the year they were born as a four-digit number. The input function always returns a string value, so use the int built-in function to convert the year value entered to an integer data type and subtract the year entered from the current year (e.g., 2019). Print a message to the user stating the value that they entered and their calculated age.

a. Code here:

```
user_input = int(input("Enter a 4-digit year of birth"))
result = int(2025) - user_input
print(f"You are {result} years old." )
```

b. Why might the calculated age not be correct and what can be done to address this:

**It does not address the month and/or day of birth which can make the answer vary +- 6 months off.**

c. If `birthyear` is a variable used to store the converted input, what's wrong with the expression `"Your age is " + birthyear`:

**The addition sign is not needed, squiggle brackets around birthyear will be required.**
**Rewrite to the following:**
**Print(f"Your age is {birthyear}.")**

d. Explain why the names of the variables are good or not good and explain why good naming is important:

**Naming provides easy reference to information or values that will be used throughout a program. Constants are a good example. Naming can be challenging because if not done**

**correctly you may find yourself lost in finding what data is what. If you are far along in a program and you wish to rename the variables, you will have to alter every instance that the named variable occurs. (this is very time consuming if it is a large code.**

2.3 Write Python code that uses the input built-in function to ask the user to enter a decimal formatted number between 1 and 100. The input function always returns a string value, so use the float built-in function to convert the value entered to a float data type and square the number that the user entered using the exponentiation operator. Print a message to the user stating the value that they entered and the square of the value that they entered.

a. Code here:

```
user_input = input("Enter a decimal-number between 1-100.")
number = float(user_input)
square = number ** 2
print(f"{user_input} squared is {square} ")
```

b. Research alternative ways to square the inputted value that do not use the exponentiation operator and explain why you may want to use such alternatives:

**you can simply take the square line to say something along the lines of**

```
square = number * number
```

c. Add or adjust comments to your code. Explain why you added these and how they are important/useful:

**Line 1**

```
user_input = input("What decimal number between 1-100 do you want to square?")
```

**Makes it more inviting and stated end goal so the user does not ask themselves "what for?"**

2.4 Modify the code in Exercise 2.3 to round the values reported to the user to two decimal places (use the round built-in function).

a. Before you write the code, look up some documentation on the round() function. What data type is the return value and why is it this data type? How do you know this? Do you think another data type might be better?

**The result is a float because the input is converted to a float and arithmetic operations involving floats return floats. Formatting the float (e.g., `.2`) is ideal for displaying results with two decimal points. If more precision or control over rounding is needed, the decimal class is a better option.**

b. Code here:

```
user_input = input("What decimal number between 1-100 do you want to square?")
number = float(user_input)
square = number * number
print(f"{user_input} squared is {round(square , 2)} ")
```

2.5 Write Python code that uses the input built-in function to ask the user to enter a sentence of their choosing. Use the `len` built-in function to determine how many characters were in the string entered and report this information back to the user.

a. Code here:

```
user_input = input("Write a sentence of your choosing.:")
character_count = int(len(user_input))
print(f"Your Sentence had {character_count} characters in it.")
```

b. Describe a situation in which it would be useful to have the length of a string

**When size requirements are important for analysis. Or restrictions are in place from user inputs. It can allow an if/then protocol to hinder the submission of a form if it does not meet or exceeds X characters.**

2.6 Write Python code that uses the input built-in function to ask the user to enter a weight in pounds. The input function always returns a string value, so use the float built-in function to convert the value entered to a float data type and determine the equivalent weight in kilograms (you can use the conversion factor that 1 pound = 0.453592 kilograms). Print a message to the user stating the weight in pounds that they entered and the equivalent weight in kilograms.

a. Write the program using one line of code. Code here:

```
print(f"You would weigh {float(input('Enter a weight in lbs: ')) * 0.453592} kilograms in Europe.")
```

b. Discuss the pros and cons of implementing it in one line:

**Pros is that it Saves space. And you could perform large computations using single lines rather than 6+ for simple equations and reproductions.**

**Cons is that if it's a complicated problem set it can be very very long. And no one will want to go through 150+ characters on a single line to visualize the problem if there needs to be any tweaking/adjustment etc.**

2.7 Write Python code that uses the input built-in function to ask the user to enter a temperature in the Fahrenheit temperature scale. The input function always returns a string value, so use the float built-in function to convert the value entered to a float data type and determine the equivalent temperature in the Celsius temperature scale (use the conversion factor °C = (°F – 32) × (5/9)). Print a message to the user stating the temperature in Fahrenheit that they entered and the equivalent temperature in Celsius. You can verify that your code executes properly by entering in 32°F (equivalent is 0°C) and 212°F (equivalent is 100°C).

a. Code here:

```
user_input = int(input("Input a temperature in Fahrenheit :"))
celcius = round((user_input - 32) * (5 / 9) , 0)
print(f"Your temperature of {user_input} degrees F. is equal to {celcius}
degrees C.")
```

b. Why do you want to verify your code using the input 32°F? What is it called when you do this kind of verification?

**Verification by zeroing**

c. Are there other verifications that should be done? Explain:

**No, it would be redundant for this simple process.**

d. Re-write the program as a function and test the function:

```
def fahrenheit_to_celsius():
    user_input = int(input("Input a temperature in Fahrenheit:"))
    celcius = round((user_input - 32) * (5 / 9) , 0)
```

```
    print(f"Your temperature of {user_input} degrees F. is equal to {celcius}
degrees C.")

fahrenheit_to_celsius()
fahrenheit_to_celsius()
fahrenheit_to_celsius()
```

a. Explain the pros and cons of creating a function to do the conversion:

**It allows a user to reuse the same process without recoding the entire calculation. Speeds up workflow and allows for less noise. If there are numerous times where you need it you can see I put it three times in a row, say it was for a valve systems that regulates heating signatures. Each input from a F sensitive sensor could then output a C degree. (if you're in a country that uses C) other applications I can't think of at the moment other than a weather app. Changing the formatting if desired by the user.**