cis122 Project 4

Define functions

Define your functions before your program calls them.

Define each function just once, even if your program calls the function 3 times.

30 points total

When you finish, **save** your notebook (click the "diskette" save icon on the far left),
test your project; make sure each part works correctly,
then go into **Canvas**, **Assignments**, **Project 4**, and **Upload** your **project4.ipynb** file to Canvas.
Click **Submit**.

Short functions can do useful things.

Example

```
def input_float(request_string):
    ''' Ask user to type a number
        Convert result to a float
        Returns floating point result
    '''
    user_string = input(request_string)
    result = float(user_string)
    return result

# Call the function
ask_for_gas_price = "Type price per gallon "
price_per_gallon = input_float(ask_for_gas_price)
print("Gas costs $", price_per_gallon, "per gallon in Oregon")
```

Part A Define some functions

1) *2 points* Get input data, convert to a whole number (int or integer)

```
def input_int(request_string):   # You fill in the rest of the function definition
```

```
# ================================================
# Do not change this - it will call your function
#
prompt = "How many sea gulls did you see today? "
seagulls = input_int(prompt)
print("You saw", seagulls, "seagulls today")

computer_count = input_int("How many laptops computers do you own? ")
print("You own", computers_count, "laptop computers")
```

2) *2 points* **Get input data, convert to a float**

```
def input_float(request_string):   # You fill in the rest of the function definition




# ================================================
# Do not change this - it will call your function
#
price_prompt = "How much does a gallon of gas cost today? "
gas_price = input_float(price_prompt)
print("Gas costs $", gas_price, "per gallon today")

latte_prompt = "How much does a Starbucks 20 ounce latte cost? "
latte_cost = input_float(latte_prompt)
print("Starbucks charges about", latte_cost, "for a large latte")
```

3) *2 points* **Convert inches to feet**

Given a number of inches, the function returns the number of feet
Given 12 inches, it should return 1.0

```
def inches_to_feet(inches):   # You fill in the rest of the function definition




# ================================================
# Do not change this - it will call your function
#
number_of_inches = 36
number_of_feet = inches_to_feet(number_of_inches)

my_distance = 11.8  # inches
my_feet = inches_to_feet(my_distance)

print(number_of_inches, 'inches =', number_of_feet, 'feet')
print(my_distance, 'inches =', my_feet, 'feet')
```

4) *2 points* **Convert feet to inches**

Given a number of inches, the function returns the number of feet
Given 12 inches, it should return 1.0

```
def feet_to_inches(feet):   # You fill in the rest of the function definition



number_of_feet = 5
number_of_inches = feet_to_inches(number_of_feet)

distance = 10.5  # feet
distance_in_inches = feet_to_inches(distance)

print(number_of_inches, 'inches =', number_of_feet, 'feet')
print(distance_in_inches, 'inches =', distance, 'feet')
```

4) *2 points* **Convert temperature Celsius to Fahrenheit**

Given a temperature in degrees Celsius, return the Fahrenheit temperature.

A few facts
100 degrees C (boiling) is 212 degrees F.
0 degrees C (freezing) is 32 degrees F.
Given 20 degrees C, you should get 68 degrees F.

```
def C_to_F(C_degrees):    # You fill in the rest of the function definition
```

```
# ================================================
# Do not change this - it will call your function
#
C_temperature = 20
F_degrees = C_to_F(C_temperature)
print(C_temperature, 'degrees C =', F_degrees, 'degrees F')
```

5) *2 points* **Convert temperature F to Celsius**

Given a temperature in degrees Fahrenheit, return the Celsius temperature.

Given 68 degrees F, you should get 20 degrees C.

```
def C_to_F(C_degrees):    # You fill in the rest of the function definition
```

```
# ================================================
# Do not change this - it will call your function
#
C_temperature = 20
F_degrees = C_to_F(C_temperature)
print(C_temperature, 'degrees C =', F_degrees, 'degrees F')
```

6) *2 points* **Convert miles to kilometers**

Given a number of miles, the function returns the number of kilometers
Given 3.1 miles, it should return around 5.0 kilometers

```
def miles_to_km(miles):    # You fill in the rest of the function definition
```

```
# ================================================
# Do not change this - it will call your function
miles_run = 3.1
km_run = miles_to_km(miles_run)

print(miles_run, 'miles =', km_run, 'km')
```

7) *2 points* Convert kilometers to miles

Given a number of kilometers, the function returns the number of miles
Given 5.0 kilometers, it should return around 3.1 miles

```
def km_to_miles(km):    # You fill in the rest of the function definition




# ================================================
# Do not change this - it will call your function
kilometers = 5.0
miles = km_to_miles(kilometers)

print(miles, 'miles =', kilometers, 'km')
```

7) *4 points* Filter data to a new list, get its total

You need to get the total of all radiation **levels 1.0 or larger**.
Use the "filter to a new list" strategy:

Your program will create a `high_radiation` list.
Iterate through the radiation_levels list.
If the item is 1.0 or more, append it to your `high_radiation` list.

After your iteration for loop ends, compute and print the sum of the `high_radiation` list.
Also print the number items in the high list (hint: len( ) function can help here).

```
radiation_levels = [0.2, 0.4, 7.6, 1.1, 0.98, 4.7, 0.5, 3.2, 4.8, 0.2]
```

8) *4 points* Filter data to a new list, get its total

Use the same list as in the problem above.
You need to get the total of all radiation levels **less than 1.0**.
Use the "filter to a new list" strategy:

Your program will create a `low_radiation` list.
Iterate through the radiation_levels list.
If the item is less than 1.0, append it to your `low_radiation` list.

After your iteration for loop ends, compute and print the sum of the `low_radiation` list.
Also print the number items in the low list (hint: len( ) function can help here).

```
radiation_levels = [0.2, 0.4, 7.6, 1.1, 0.98, 4.7, 0.5, 3.2, 4.8]
```

9) *5 points* Define a function grade
Given a points score, it returns a letter grade 'A', 'B', 'C', 'D' or 'F'.

Apply these rules in order to figure out the grade in the Prineville Academy.

If the score is **90** or better, it's a grade '**A**' (skip past all remaining tests).

Else if the score is **80** or better, it's a grade '**B**' (skip past all remaining tests).

Else if the score is **75** or better, it's a grade '**C**' (skip past all remaining tests).

Else if the score is **65** or better, it's a grade '**D**' (skip past all remaining tests).

Else it's an '**F**'.

```
# Python has a way to handle else if -- the elif keyword (sort of mashes else and if together).
if some condition:
    # indented block
    # at end of block, skips past all remaining elif and else's.
elif some condition:
    # indented block
    # at end of block, skips past all remaining elif and else's.
elif some condition:
    # indented block
    # at end of block, skips past all remaining elif and else's.
...
else:
    # indented block
# all paths rejoin here at first unindented statement after else

def get_grade(score):   # You fill in the rest of the function definition
```

```
# ==============================
# Test your function
# Repeat some tests
#     Ask for points earned on an assignment
#     (Be sure to convert the answer to a number)
#     Call the get_grade function giving it the points
#     print the points and the letter grade
```

Among the assignment points you use to test your program with,
include points like 100, 99, 90, 89.9, 80, 22, 75, 72.

10) *5 points* **Define a function leap_year**
It returns 1 when the year such as 2008 is a leap year,
0 otherwise.

Generally, the rules are simple. Years divisable by 4 is a leap year, other years are not.
But century years (divisible by 100) are not leap years unless the century is divisble by 400.
You can use these rules for years since around 1600 in Western Europe and since 1917 in Russia.

Let's list the rules in a way that makes it all easier to figure out how to compute a leap year value.

Python's remainder operator **%** can tell you if an integer is divisible by another integer.

```
a = 6
b = 3
remainder = a % b
if remainder == 0:
    print(b, 'divides', a)
else:
    print(b, 'does not divide', a, 'remainder is', remainder)
#
```

1) If the year is divisible by 400, it's a leap year (set leap_year to 1).

2) Else if the year is divisible by 100, it's **not** a leap year (set leap_year to 0).

3) Else if the year is divisible by 4, it's a leap year (set leap_year to 1).

4) Else it's not a leap year (set leap_year to 0).

```
def leap_year(year):   # You fill in the rest of the function definition



# Some tests to run to see if leap year is defined correctly.
year = 1900
leap = leap_year(year)
if leap == 1:
    print(year, 'is a leap year')
else:
    print(year, 'not a leap year')

year = 1977
leap = leap_year(year)
if leap == 1:
    print(year, 'is a leap year')
else:
    print(year, 'not a leap year')

year = 2000
leap = leap_year(year)
if leap == 1:
    print(year, 'is a leap year')
else:
    print(year, 'not a leap year')

year = 2004
leap = leap_year(year)
if leap == 1:
    print(year, 'is a leap year')
else:
    print(year, 'not a leap year')
```

Challenges - each 1 point

Try some function definitions that apply for your own major.
A business finance major, for example, might need a function to calculate the **future value** of an investment.
An economist might have access to some **GDP** figures that need analysis.
Try creating some filters and functions that relate to your interest or major.

Or try defining a function such as a car's **gas mileage** computation. It would take 2 parameters, distance traveled and number of gallons used, returning the miles per gallon computation, suitably rounded.

GTF challenge Your GTF may also offer 1 or 2 point challenges as well.

Practice makes perfect.

Remember to try pythontutor.com if you hit snags.