Practice

Content Learning Objectives

- Practice with everything
- Some basic arithmetic
- if/elif/else statements
- for loops
- lists

Task

Below there are a few functions that will complete a specific task. The function definition will be given to you with the parameters unique to that function.

Follow the requirements listed to write the complete function. Once you are done writing the function, run the provided test cases to see if your function works correctly.

Function 1: Atmospheric Stability

When an air parcel temperature is compared to the environmental temperature you can determine its stability.

If the parcel temperature is cooler than or equal to the environmental temperature, the air is stable.

If the parcel temperature is warmer than the environmental temperature, the air is unstable.

Requirements

This function shall:

- use an if block
- determine if the air is stable or unstable
- print whether the air is stable or unstable

In [ ]:
```
def stability(parcel_temp, environmental_temp):
    #code goes here
    if parcel_temp <= envirnmental_temp:
       print ( “ air is stable ” )
    else:
       parcel_temp > enviornment_temp
       print ( “ air is unstable ” )

    def main ():
      parcel_temp = 14
      enviornmental_temp = 20
      stability(parcel_temp, enviornmental_temp)
    main()
```

```
stability(14, 20)
stability(19,17)
stability(21,21)
```

Function 2: ENSO Classification

There are natural variations in the sea surface temperature (SST) of the Pacific Ocean. As the SST changes, it can have effects on global weather patterns through the Oceanic Niño Index (ONI). If the Pacific Ocean is greater than or equal to 0.5 ^{o}C warmer than usual then it is an El Niño year. If the Pacific Ocean is greater than or equal to 0.5 ^{o}C cooler than usual then it is a La Niña year. Otherwise, it is an ENSO Neutral year.

El Niño years are characterized by milder wetter winters in the northeast US, wetter conditions in the south east US, and warmer temperatures in the Pacific North West and Midwest.

La Niña years are charcterized by cooler temperatures in the Midwest, wetter weather in the Pacific North West, and drier weather in the south.

ENSO Neutral years are charcterized by average weather throughout all of the US.

Requirements

This function shall:

- use an if/elif/else statement
- determine if a given year is an El Niño, La Niña or ENSO Neutral Year
- print the expected weather conditions across the US

In [ ]:
```
def ensoClassification (SST):
    # code goes here
      if SST >= 0.5:
         print ("El Nino: Expect wetter winters in the northeast US, wetter
      conditions in the south east US, and warmer temperatures in the Pacific
      North West and Midwest.")

      elif SST <= -0.5:
         print ("LaNina: Expect cooler temperatures in the Midwest, wetter
      weather in the Pacific North West, and drier weather in the south.")
      else:
         print ("ENSO Neutral year: Expect average weather throughout all of
      the US. ")

      def main():
         SST = 12
         ensoClassification(SST)
      main()
```

```
ensoClassification(0.4)
ensoClassification(0.9)
ensoClassification(-1.2)
ensoClassification(-0.3)
```

Function 3: Winds

Winds naturally change speed and direction on hourly and daily time scales. In this case, wind direction is given as a string (N, E, S, W) and wind speed is given in kts.

Requirements

This function shall:

- convert wind direction to a number
- calculate the average wind speed
- calculate the average wind direction
- return the average wind speed and direction

In [ ]:
```
dir1 = [12,5,18,9,22,14,7,16,3,11,25,6,19,10,8,2 ]
speeds1 = [ 'N','E', 'W','S', 'N', 'E','S', 'W','N',
            'E', 'W','S', 'E', 'N','W']

dir2 = [5, 8, 12, 6, 14, 9, 20, 11, 7, 16, 10, 18, 22,
        13, 4, 15, 19]
speeds2 = ['N', 'E', 'S', 'W', 'N', 'E', 'S', 'W', 'N',
           'E', 'S', 'W', 'N', 'E', 'S', 'W', 'N']

dir3 = [3, 7, 11, 9, 17, 6, 13, 8, 15, 5, 19, 12, 10,
        4, 14, 16, 18]
speeds3 = ['W', 'S', 'E', 'N', 'W', 'E', 'S', 'N', 'E',
           'W', 'S', 'N', 'E', 'S', 'W', 'N', 'E']



def avg_winds(directions, speeds):

    total = 0
    for i in range(len(speeds)):
       if speeds [i] == "N":
          speeds [i] = 360
       elif speeds [i] == "E":
          speeds [i] = 90
       elif speeds [i] == "S":
          speeds [i] = 180
       elif speeds [i] == "W":
          speeds [i] = 270
    total = 0
    for x in range(len(directions)):
       total = total + directions [i] # or total += directions [i]
    average_speed= total / len(directions)

    total = 0
    for x in range(len(speeds)):
       total = total + speeds [i] # or total += directions [i]
    average_direction= total / len(speeds)









    return average_speed, average_dir


print(avg_winds(dir1,speeds1))
print(avg_winds(dir2,speeds2))
print(avg_winds(dir3,speeds3))
```

Function 4: Relative Humidity

Relative Humdity can be calculated by dividing the actual vapor pressure by the saturation vapor pressure and multiplying by 100 to get a percent.

This function shall:

- calculate relative humidity
- return the calculated relative humidity

In [ ]:
```
def RH_calc(vap_press, sat_vap_press):
    relative_humidity =  (vap_press / sat_vap_press) * 100
    return retalive humidity
    return relative_humidity

print(RH_calc(24, 37))
print(RH_calc(13, 13))
print(RH_calc(20.1,30))
```

Function 5: Potential Temperature

Potential temperature is an estimate for the temperature an air parcel would have if it was moved adiabatically from aloft to a standard pressure, typically 1000 mb. It can be calculated with this equation: $ θ = temp * (starting pressure / 1000) ^{0.286} $. As an Atmospheric Sciences student, you will explore potential temperature in much more depth later.

This function shall:

- be named potential_temperature
- have two parameters (temp and press)
- calculate potential temperature
- return calculated potential temperature (pot_temp)

In [ ]:
```
# write function here
def potential_temperature (temp,press):
    poten_temp = temp*(press/1000)**0.286
     return poten_temp

#make sure to return a value


potential_temperature(-20, 300)
potential_temperature(-14, 425)
potential_temperature(6, 575)
```

In [ ]:
```

```