Stats 21: Homework 1

Menh Phuong Nguyen

Acknowledgements: several of these problems are copied from or modified from Think Python by Allen Downey.

I've started the homework file for you. You'll need to fill in the rest with your answers. My encouragement is to use the keyboard shortcuts as much as possible and use the mouse as little as possible while working the Jupyter Notebook.

After you complete the homework with your answers, go to the menu and choose **Kernel > Restart & Run All.** Review the document to **make sure all requested output is visible**. You will not get credit for problems where the requested output is not visible, even if the function you coded is correct.

When you are satisfied with the output, choose File > Download As ... > PDF or HTML. If you choose to save as HTML, you'll then need to "Print as PDF". Submit the PDF to Gradescope.

Submit this ipynb file, complete with your answers and output to Canvas / Bruin Learn.

Again, you must make sure all requested output is visible to receive full credit.

Task 1

Create an account on GitHub.

Change your profile picture. Ideally, use a photo of yourself that would be appropriate for a resume. If you are not comfortable with the idea of using a photo of yourself, use any other image that is suitable for a workplace environment.

Follow the instructions provided in class to fork the class repository to your GitHub.

Create another repository with at least two text files in it on GitHub (other than the forked class notes repository). Make at least two additional commits to the repository and push them to GitHub.

Provide a link to both repositories here.

You will also need to submit the link to your own repository (not the forked one) to Canvas / Bruin Learn.

Your Answer:

- Link to your forked repository: https://github.com/Menh-Phuong-Nguyen/Stats-21-Class-Repository
- Link to your own repository: https://github.com/Menh-Phuong-Nguyen/Stats-21

Problem 2

An important part of programming is learning to interpret error messages and understanding what correction needs to be made.

Read and familiarize yourself with the following error messages.

Explain the error. Then duplicate each cell and correct the error. The first problem has been done for you as an example.

In [1]:
```
# A
print("Hello World"
```

```
  Cell In[1], line 2
    print("Hello World"
                       ^
SyntaxError: incomplete input
```

Answer: The `print()` function is missing the closing parenthesis. This results in an unexpected EOF error.

In [2]:
```
# A [corrected]
print("Hello World")
```

```
Hello World
```

In [3]:
```
# B
print("Hello")
    print("Goodbye")
```

```
  Cell In[3], line 3
    print("Goodbye")
    ^
IndentationError: unexpected indent
```

Answer: Indentations are used to introduce and define a new block statement. Indenting in the above scenario does not make much sense. We simply want to make a call to a function and are not defining a function ourselves.

In [4]:
```
# B [corrected]
print("Hello")
print("Goodbye")
```

```
Hello
Goodbye
```

In [5]:
```
# C
x = 10
```

```
if x > 8
    print("x is greater than 8")
```

```
  Cell In[5], line 3
    if x > 8
            ^
SyntaxError: expected ':'
```

Answer: The conditional of an `if` statement must be followed by a colon `:` to signify the start of the body statements for the `if` statement.

In [6]:
```
# C [corrected]
x = 10
if x > 8:
    print("x is greater than 8")
```

```
x is greater than 8
```

In [7]:
```
# D
if x = 10:
    print("x is equal to 10")
```

```
  Cell In[7], line 2
    if x = 10:
       ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Answer: A single equals `=` denotes assignment **not** comparison! For the conditional, we want to compare the variable `x` with the value `10` which calls for using double equals `==`.

In [8]:
```
# D [corrected]
if x == 10:
    print("x is equal to 10")
```

```
x is equal to 10
```

In [9]:
```
# E
x = 5
if x == 5:
print("x is five")
```

```
  Cell In[9], line 4
    print("x is five")
    ^
IndentationError: expected an indented block after 'if' statement on line 3
```

Answer: As mentioned above, indentations are used to introduce and define a new block statement. Use of `if` statements introduces a new scope / block statement. Rather than using `{}` as one would in other languages, we use indentations to produce the same results.

In [10]:
```
# E [corrected]
x = 5
if x == 5:
    print("x is five")
```

```
x is five
```

In [11]:
```
# F
l = [1, 2, 50, 10]
l = sort(l)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 3
      1 # F
      2 l = [1, 2, 50, 10]
----> 3 l = sort(l)

NameError: name 'sort' is not defined
```

Answer: The `sort` function is a method that belongs to the `list` object and therefore must be called using the following format *listname.methodname()*.

In [12]:
```
# F [corrected]
l = [1, 2, 50, 10]
l = l.sort()
```

Problem 3

Use Python as a calculator. Enter the appropriate calculation in a cell and be sure the output value is visible.

A. How many seconds are there in 42 minutes 42 seconds?

In [13]:
```
42 * 60 + 42
```

Out[13]:
```
2562
```

B. There are 1.61 kilometers in a mile. How many miles are there in 10 kilometers?

In [14]:
```
10 / 1.61
```

Out[14]:
```
6.211180124223602
```

C. If you run a 10 kilometer race in 42 minutes 42 seconds, what is your average 1-mile pace (time to complete 1 mile in minutes and seconds)? What is your average speed in miles per hour?

In [15]:
```
print(Out[13] / Out[14])
print(Out[14] / (Out[13] / 3600))
```

```
412.482
8.727653570337614
```

Problem 4

Write functions for the following problems.

A. The volume of a sphere with radius r is

$$V = \frac{4}{3}\pi r^3$$

Write a function `sphere_volume(r)` that will accept a radius as an argument and return the volume.

- Use the function to find the volume of a sphere with radius 5.
- Use the function to find the volume of a sphere with radius 15.

In [16]:
```
import math
def sphere_volume(r):
    return 4 / 3 * math.pi * pow(r, 3)

print(sphere_volume(5))
print(sphere_volume(15))
```

```
523.5987755982989
14137.166941154068
```

B. Suppose the cover price of a book is $24.95, but bookstores get a 40\% discount. Shipping costs \$3 for the first copy and 75 cents for each additional copy.

Write a function `wholesale_cost(books)` that accepts an argument for the number of books and will return the total cost of the books plus shipping.

- Use the function to find the total wholesale cost for 60 copies.
- Use the function to find the total wholesale cost for 10 copies.

In [17]:
```
def wholesale_cost(books):
    price = 24.95 - (24.95 * 0.4)
    if (books > 0):
        price += 3
        books -= 1
    while (books > 0):
        price += 0.75
        books -= 1
    return price

print(wholesale_cost(60))
print(wholesale_cost(10))
```

```
62.22
24.72
```

C. A person runs several miles. The first and last miles are run at an 'easy' pace. Other than the first and last miles, the other miles are at a faster pace.

Write a function `run_time(miles, warm_pace, fast_pace)` to calculate the time the runner will take. The function accepts three input arguments: how many miles the runner travels (minimum value is 2), the warm-up and cool-down pace, the fast pace. The function will print the time in the format minutes:seconds, and will return a tuple of values: (minutes, seconds)

Use the function to find the time to run a total of 5 miles. The warm-up pace is 8:15 per mile. The speed pace is 7:12 per mile.

Call the function using: `run_time(miles = 5, warm_pace = 495, fast_pace = 432)`

In [18]:
```python
def run_time(miles, warm_pace, fast_pace):
    time = warm_pace * 2
    while (miles > 2):
        time += fast_pace
        miles -= 1

    minutes = time // 60
    seconds = time % 60

    print(str(minutes) + ":" + "{:02d}".format(seconds))
    return minutes, seconds

run_time(miles = 5, warm_pace = 495, fast_pace = 432)
```

```
38:06
```

Out[18]:
```
(38, 6)
```

Another important skill is to be able to read documentation.

Read the documentation for the function str.split() at
https://docs.python.org/3/library/stdtypes.html#str.split

Adjust the function so that the call can be made with minutes and seconds:

`run_time(miles = 5, warm_pace = "8:15", fast_pace = "7:12")`

In [19]:
```python
def run_time(miles, warm_pace, fast_pace):
    minutes = int(warm_pace.split(":")[0]) * 2
    seconds = int(warm_pace.split(":")[1]) * 2

    fast_min = int(fast_pace.split(":")[0])
    fast_sec = int(fast_pace.split(":")[1])

    while (miles > 2):
        minutes += fast_min
        seconds += fast_sec
        miles -= 1

    secs_to_mins = seconds // 60
    seconds = seconds % 60
    minutes += secs_to_mins

    print(str(minutes) + ":" + "{:02d}".format(seconds))
    return minutes, seconds
```

```
run_time(miles = 5, warm_pace = "8:15", fast_pace = "7:12")
```

```
38:06
Out[19]: (38, 6)
```

Problem 5

Use `import math` to gain access to the math library.

Create a function `polar(real, imaginary)` that will return the polar coordinates of a complex number.

The input arguments are the real and imaginary components of a complex number. The function will return a tuple of values: the value of the radius `r` and the angle `theta`.

For a refresher, see: https://ptolemy.berkeley.edu/eecs20/sidebars/complex/polar.html

Show the results for the following complex numbers:

- 1 + i
- -2 - 3i
- 4 + 2i

In [20]:
```
import math
def polar(real, imaginary):
    r = math.sqrt(pow(real, 2) + pow(imaginary, 2))
    theta = math.atan(imaginary / real)
    return r, theta

print(polar(1, 1))
print(polar(-2, -3))
print(polar(4, 2))
```

```
(1.4142135623730951, 0.7853981633974483)
(3.605551275463989, 0.982793723247329)
(4.47213595499958, 0.4636476090008061)
```

Problem 6

Define a function called `insert_into(listname, index, iterable)`. It will accept three arguments, a currently existing list, an index, and another list/tuple that will be inserted at the index position.

Python's built-in function, `list.insert()` can only insert one object.

In [21]:
```
def insert_into(listname, index, iterable):
    while (len(iterable) > 0):
        listname.insert(index, iterable[len(iterable) - 1])
        iterable.pop()
```

```
In [22]: # do not modify. We will check this result for grading
         l = [0,'a','b','c',4,5,6]
         i = ['hello', 'there']
         insert_into(l, 3, i)
```

Problem 7

Define a function called `first_equals_last(listname)`

It will accept a list as an argument. It will return `True` if the first and last elements are equal and if the list has a length greater than 1. It will return False for all other cases.

```
In [23]: def first_equals_last(listname):
             return len(listname) > 1 and listname[0] == listname[len(listname) - 1];
```

```
In [24]: # do not modify. We will check this result for grading
         a = [1,2,3]
         first_equals_last(a)

Out[24]: False
```

```
In [25]: # do not modify. We will check this result for grading
         b = ['hello','goodbye','hello']
         first_equals_last(b)

Out[25]: True
```

```
In [26]: # do not modify. We will check this result for grading
         c  = [1,2,3,'1']
         first_equals_last(c)

Out[26]: False
```

```
In [27]: # do not modify. We will check this result for grading
         d = [[1,2],[3,2],[1,2]]
         first_equals_last(d)

Out[27]: True
```