HW1_Notebook

April 6, 2021

1 Stats 21: Homework 1

1.1 Ivan Tran

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. Go through 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.

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

2 Task 1

Create an account on GitHub.

Change your profile picture. Ideally, use 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.

Create a repository on GitHub (other than the forked class notes repository). Make at least two additional commits to the reposiotry and push them to GitHub.

Provide a link to your repository here.

2.1 Your Answer:

Link to your repository: https://github.com/ivantran96/Stats_21-IT

3 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.

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

```
  File "<ipython-input-1-41ea49db4490>", line 2
    print("Hello World"
                       ^
SyntaxError: unexpected EOF while parsing
```

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

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

```
Hello World
```

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

```
  File "<ipython-input-3-5488ccf53c57>", line 3
    print("Goodbye")
    ^
IndentationError: unexpected indent
```

Answer: Indentation is used to mark block codes. There is an error since the second `print()` statement is not in the same block code as the first statement.

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

```
Hello
Goodbye
```

[5]:
```
x = 10
if x > 8
    print("x is greater than 8")
```

```
  File "<ipython-input-5-f4188b1e80d6>", line 2
    if x > 8
            ^
```

```
SyntaxError: invalid syntax
```

Answer: `:` is missing at the end of the if statement.

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

```
x is greater than 8
```

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

```
  File "<ipython-input-7-5d76e710ef1a>", line 1
    if x = 10:
         ^
SyntaxError: invalid syntax
```

Answer: `==` is supposed to be used instead of `=` when comparing values.

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

```
x is equal to 10
```

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

```
  File "<ipython-input-9-abab10e283c7>", line 3
    print("x is five")
    ^
IndentationError: expected an indented block
```

Answer: An indentation is expected after the if statement. Since the line where the `print()` function is is not indented, an error is returned.

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

```
x is five
```

[11]:
```python
l = [1, 2, 50, 10]
l = sort(l)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-429480b02130> in <module>
      1 l = [1, 2, 50, 10]
----> 2 l = sort(l)

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

Answer: `sort()` is a method of list objects, so it is supposed to be attached to the end of the list object. Also, it changes the list directly, rather than the function `sorted()` which just returns the sorted list.

[12]:
```python
l = [1, 2, 50, 10]
l.sort()
```

4 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?

[13]:
```python
print(42 * 60 + 42, "seconds")
```

```
2562 seconds
```

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

[14]:
```python
print("{:.2f}".format(10 / 1.61), "miles")
```

```
6.21 miles
```

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?

[15]:
```python
# avg mile pace
avg_min = (42 * 60 + 42) / (10 / 1.61) / 60
print("Average 1-mile pace is", avg_min // 1, "minutes and", "{:.2f}".
 ↪format(avg_min % 1 * 60), "seconds.\n")
# avg speed mph
mph = (10 / 1.61) / ((42 + 42 / 60) / 60)
print("{:.2f}".format(mph), "miles per hour")
```

```
Average 1-mile pace is 6.0 minutes and 52.48 seconds.

8.73 miles per hour
```

5 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.

[16]:
```python
from math import pi

def sphere_volume(r):
    return (4/3) * pi * r**3

# radius = 5
print(sphere_volume(5))
# radius = 15
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.

[17]:
```python
def wholesale_cost(books):
    book_price = books * 24.95 * .6
    if books == 1:
        shipping = 3
    elif books > 1:
        shipping = 3 + .75 * (books - 1)
    else:
        shipping = 0
    return book_price + shipping
```

```
# 60 copies
print("$" + "{:.2f}".format(wholesale_cost(60))+ "\n")
# 10 copies
print("$" + "{:.2f}".format(wholesale_cost(10)))
```

```
$945.45

$159.45
```

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)`

[18]:
```
def run_time(miles, warm_pace, fast_pace):
    warm_time = 2 * warm_pace
    fast_time = (miles - 2) * fast_pace
    time = warm_time + fast_time
    mins = round(time / 60 // 1)
    secs = round(time / 60 % 1 * 60, 2)
    print(str(mins) + ":" + str(secs))
    return (mins, secs)

# 5 miles, warm-up pace 8:15, speed pace 7:12
run_time(miles = 5, warm_pace = 495, fast_pace = 432)
```

```
38:6.0
```

[18]:
```
(38, 6.0)
```

Another important skill is to be able to read

Now look up 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")`

[19]:
```
def run_time(miles, warm_pace, fast_pace):
    nwarm_pace = int(warm_pace.split(":")[0]) * 60 + int(warm_pace.split(":
↪")[1])
    nfast_pace = int(fast_pace.split(":")[0]) * 60 + int(fast_pace.split(":
↪")[1])
```

```
    warm_time = 2 * nwarm_pace
    fast_time = (miles - 2) * nfast_pace
    time = warm_time + fast_time
    mins = round(time / 60 // 1)
    secs = round(time / 60 % 1 * 60, 2)
    print(str(mins) + ":" + str(secs))
    return (mins, secs)

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

```
38:6.0
```

```
[19]: (38, 6.0)
```

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

```
[20]: import math
def polar(real, imaginary):
    r = math.sqrt(real**2 + imaginary**2)
    theta = math.atan(imaginary / real)
    return (round(r, 3), round(theta, 3))

# 1 + i
print(polar(1, 1), "\n")
# -2 - 3i
print(polar(-2, -3), "\n")
# 4 + 2i
print(polar(4, 2), "\n")
```

```
(1.414, 0.785)

(3.606, 0.983)
```

```
(4.472, 0.464)
```

7 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.

[21]:
```
# write your code here
def insert_into(listname, index, iterable):
    i = index - len(listname)
    for ins in iterable:
        listname.insert(i, ins)
```

[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)
```

8 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 the if the list has a length greater than 1. It will return False for all other cases.

[23]:
```
# write your function here
def first_equals_last(listname):
    if (len(listname) > 1) and (listname[0] == listname[-1]):
        return True
    return False
```

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

[24]:
```
False
```

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

[25]:
```
True
```

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

[26]:
```
False
```

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

[27]:
```
True
```