Introduction to Python

Homework Requirements

Please note: 20% of your grade will reflect your adherence to these instructions, which include notations, style and the **Five Requirements**, detailed below.

Five Requirements

1. Follow instructions carefully: *please re-read assignment just before submitting* to avoid missing any specific project requests.

2. Make sure you've completed the checklist: please review assignment checklist just before submitting. Ensure that you have successfully performed all tests as noted.

3. Notations must be made in the right margin with ***type, value*** *only!* No additional explanations, please. See Type/Value Notations, below.
    - **input()**: please put a sample value, and have succeeding values reflect this value
    - **if, elif, while**: please notate as **bool** and give the value that would occur if the program were run (Session 2 onwards)
    - **for**: please notate the type and value of the *first value* of the control variable (Session 3 onwards)
    - **print()**, **else**, **try/except**: these should not be notated

    Careless notation of type or value will impact your grade -- see Grading, below.

4. Use proper style: please follow all recommendations in the **Style Requirements** document (this is required starting with Session 2 homework).

    Careless application of style will impact your grade -- see Grading, below.

5. Make sure your program runs, or let the instructor know: **you must inform instructor (in a comment) if your program does not run properly or without correct output!** Make sure to run your program and all tests after making any changes, as any edits could easily break your code or change its output.

    Code that does not run or does not pass tests will impact your grade -- see Grading, below.

[more below]

Type / Value notations

1. All statements that result in a value should be notated with **type, value** only. However, please do not notate **print()**, **else,** statements. **Please place your notations *apart from the code aligned with a straight left edge.***

   (Also note that ***italic notes*** should not be included.)

```
word1 = 'hello'           # str, 'hello'
word2 = ' world!'         # str, ' world'

sentence = word1 + word2  # str, 'hello world!'

zz = len(sentence)        # int, 12

print(zz)                 #  (no need to notate print() statements)
```

2. For *dynamic* values that are determined at runtime (like **input()**), use a 'sample' value that you can imagine the user typing when the program is run.

```
a = 5.5                     # float, 5.5

b = input('enter a num: ')  # str, '10'  (for input(), notate a 'sample' value)

c = int(b)                  # int, 10    (this value is based on the sample value above)

d = a + c                   # float, 15.5
```

3. For *conditionals* (if/elif/else), notate as **bool** with the result you expect based on current values (as below, the num entered is **-5**, so the first **if** test is **False** and the second **True**).

```
a = input('enter num: ')    # str, -5         (notate 'sample' value)

if int(a) > 0:              # bool, False     (notate result of 'if' test)
    print('greater')

elif int(a) < 0:            # bool, True      (notate result of 'elif' test)
    print('lesser')
```

4. For *looping blocks* (**while** and **for**), some variables may be assigned many times and have many values. Please notate *just the first value each variable will hold***.**

```
# notating 'while' loops

a = 0                 # int, 0

while a < 5:          # bool, True     (notate first value encountered)

    a = a + 1         # int, 1         (notate first value encountered in a loop)
```

```
# notating ‘for’ loops with files

fh = open('revenue.csv')  # 'file' object

mysum = 0                 # int, 0

for line in mylist:       # str, "Haddad's,PA,239.50\n"  (notate first line)

    mysum = mysum + int(item)  # int, 1    (notate first value encountered)
```

Due dates and resubmits

1. Project solutions are due the night before class, anytime. Late submissions can only be accepted if you have contacted the instructor beforehand.

2. If a project solution is correct, it will be marked with a 'c' (for example, **2.1c**)

3. If a project solution is incomplete, it will marked [INCOMPLETE]. You may then resubmit the solution to score a completion. (Extra credit incompletes are not marked at all.)

4. You will have only two resubmits to complete your solution, except as determined by the instructor. After two resubmits if the solution has not been corrected, it will be marked incomplete permanently.

5. If a project solution is marked incomplete, it must be completed within two weeks of the initial due date. After that time the solution will be marked incomplete permanently, so don't wait to resubmit!

6. As noted in **Five Requirements**, above, reduction of your solutiuon grade will result if you are careless in completing any of the above -- see **Grading**, below.

Grading

Assignments are graded on a pass/fail basis. If something needs to be changed or improved I will return your solution with notes, and you may resubmit the solution with corrections for full credit. However, if any of the above **Five Requirements** are not met, or if a request or checklist item in the **homework assignment** is not met, I will return the solution and deduct a penalty of **5 percentage points** from the solution's potential score.

Again, you can resubmit for full credit (minus the penalty), but if there are still issues or corrections to be made, I will return the solution again and deduct **3 percentage points** from the solution if it appears that you are not taking proper care to do everything correctly.

In many cases I will not point out the specific item that’s missing, but simply say “review the checklist” or “make sure your notations are complete and correct” and you’ll need to go back through the assignment to discover what’s missing.

But I do want to emphasize that just because I return a solution that does not mean I'm taking off points. Sometimes I return a solution because I want to make sure you're doing things correctly or if you make a common mistake that we haven't discussed. **The point penalties are only applied for carelessness!** If you're very sincere in your efforts, you won't be penalized at all.

Note that as this is a new policy, there may be some "bumps" in the road or issues to resolve, so please understand that *I will be extremely lenient in my meting out of penalties*. **I simply want to ensure that you will read the assignment twice and fulfill all of the homework requirements.**

For you: tips on working on a solution

Below are some suggestions for proceeding to work on a solution -- these are not requirements.

1. Review the week's features we explored in the inclass exercises. You can also see the week's features in pythonreference.com – look for the 'Session' links on the left menu.

2. Do as many of the inclass exercises as you can. The problems and solutions there will demonstrate how to do the projects. They also demonstrate proper usage and style. If you can find the time to do all of them, by the time you start the homework you'll have a very good idea of what you need to do to solve it.

3. Plan your solution: consider the assignment and try to imagine how you might apply some of the week's features to the problem.

    a. Try to develop a mental picture of the data you're working with (for example, strings that come from **input()**, or lines of a CSV file) and how that data needs to be processed to get to the result.

    b. Consider what we discussed in class and in the exercises, and think about how some techniques might apply to the problem.

    c. Try to break down the problem into steps (first we have to do this, next we need to do this, etc.)

    d. You will eventually be able to develop a 'story' of how the input data is transformed into the output you need. This may not come right away, but we'll work towards it.

    e. You may begin coding the initial steps before you know the overall 'story', but you should always notate the object type and value as you work (see below) -- otherwise, you may lose touch with what the program is doing.

4. Writing your code:

    a. Write **one to three statements** in your code.

    b. Determine the result of each of these statements in one of the following ways:

i. run the program and print the values (and type if needed) of the variables that have been created or changed by the newly added statements
ii. review the feature in **pythonreference.com** to verify the type of any statement

c. In the right margin of your code, notate the type and value result of each statement.

Above all, you must not use code that you don't understand. *Understanding* means knowing the object type and value that results from each statement. Without it, you are *guessing.* Guessing can be a huge waste of your time!

5. <u>If unsure how to proceed with a plan, story or steps, please read the Project Dicusssion document</u> for a longer discussion, hints and some code examples or outline. Also please remember that the inclass exercises usually relate *directly* to what you are doing, so there are a lot of clues there that point toward what your project solution will be.

6. <u>Resist the temptation to guess</u>. It's hard, I know. I've heard from many students that it is a challenge to keep from "just trying something" to "see what will result if I change this". But, this won't be helpful to your learning unless you understand what the code is doing. The correct way to work is to ask what is needed, and then to look for the feature that meets that need, rather than using a feature and seeing if the result "looks right". This isn't understanding -- it's guessing! (I should add that a little guessing is ok, but once you see the result you must test it for type and value so you can see exactly what was the result!)