Lab 2: Simple to advanced coding

Objectives

- Iterative program development
- Handling syntax/semantics errors
- Writing readable code
- Writing flexible and configurable code
- Modular programming
- Documentation and commenting
- Testing and debugging

Preliminary Setup

1. You made a folder, i.e., your **workspace,** on your computer where you work on both projects and labs for this course. Each of the projects and labs will be **folders inside the workspace folder**.

PyCharm

Launch PyCharm. Create a New Project.

Note: before you click Create, make sure the project is setup the way we want:

1. Its location should be inside the workspace folder you created.
2. It should be using a system Python interpreter (**not a virtual environment**).
3. It should be using a version of Python 3 (preferably > 3.7)
4. Then change the location of the project by typing in the Location box.
5. Do not use the New Environment using virtual environment.
6. We can fix this by selecting Existing Interpreter and choosing the Python 3 interpreter we have installed. If there are no Python Interpreters to use, you might need to install one. See python.org

Add some Starter Code

Copy/paste or add change_me_fill.py to your project folder. It is at the end of this file and on Nexus.

Have a look around

Now that you have a project and some code, explore to see what you have:

1. Read the code. Can you figure out what each line is doing? If there are functions you don't know, look up the at the documentation official python documentation site.

Try to Run the code. Is there a problem? If so, try to fix the problems by following the instructions provided as comments. Make each of these additions separately, one at a time. Comment out the previous one each time you complete that step. After each one, test that your program works. *Do not try to jump in and make all these changes at once.* Write simple comments that explains what was changed for each part.

How to turn in this lab

Before turning in any program in this class, remember this mantra:

Just because it works does not mean it's good.

Your grade will also come from the following aspects of your code:

- Submission
- Accuracy/correctness
- Readability
- Neatness
- Presentation
- Style
- Testing
- Commenting

For all labs, turn in only an **electronic** version.

Please submit the followings after all labs:

- zip file of your project (the project folder, not just the .py file(s)): zip file name will be your **YourFirstNameLastName.zip**
- a single pdf file of all your codes (.py), screenshots of your output for each step: pdf file name will be **YourFirstNameLastName.pdf**

Submit

- **the zip and pdf files**
- after Monday Lab session until Tuesday 8 AM EST
- from the Nexus Lab submission link that will be accessible in Week 2-10

Ask for help if you are having problems!

change_me_fill.py

```
# I am calculating n factorial, but I need some changes.
# Is there any syntax/naming/logical error?
# Is the code readable?
# FIXME
# Copy the following code and change it step by step
# Comment out the previous one each time you complete that step.
# Step 0
# Original code
n = 5
for i in range(n):
    1x *= i
print(1x)


# Step 1
# Fix the naming error(s) and undefined names in the above code
# Fix naming errors and initialize them
# Fix range values


# Step 2
# Names like x, i are not self-descriptive
# Rename your variables so that they are self-descriptive
# Use Refactor --> Rename


# Step 3
# The code is not flexible and configurable
# Change the code so that n value is an input
# Do not forget to convert input function return value to an int
# Do not forget to provide a meaningful message for the input function


# Step 4
# It is time to make this code modular
# It is a good idea to make this code a function
# Your function will be named accordingly
# it has a parameter of type int and return an int
# Convert the code to a function
# Use type hinting
# Use docstring reStructured Text style


# Call your function for a user input and print the result


# Step 5
# Calling the function for a single input is not good enough
# Test your function with various inputs
# You may use a for loop to get k inputs and test your function



# Step 6
# Calling the function for random inputs is not good enough
# Test your function with various inputs, like negative, positive, zero
# You may use a for loop ranging from negatives to positives



# Step 7
# Modify your function so that it will print an error message return 0 for negative
numbers


# Step 8
# Test your function with various inputs, like negative, positive, zero
# You may use a list of negatives, zero and positives
```