**Starter Code**

* [**https://repl.it/@cloudtimeskong/BonyWellwornSquares#main.py**](https://repl.it/@cloudtimeskong/BonyWellwornSquares#main.py)
* **Milestone 1 Solutions**
  + [**https://repl.it/@cloudtimeskong/CheerySparklingAudacity#main.py**](https://repl.it/@cloudtimeskong/CheerySparklingAudacity#main.py)
  + **Updated on November 17th** 
    - **Added two new functions for hint 4**
* **Add to this code**
  + **Do not start from scratch**
* **For milestone 2 , work with milestone 1 solutions**

**Introduction**

You will implement an increasingly complicated calculator program

**Milestone #1-November 13th 11:59pm**

1. Request the user for an operation to perform a calculation in a string
   1. Inputted numbers will always be positive
2. Create relevant functions where needed
3. Finish Implementing subtraction
4. The calculator can perform a simple operation using only one operator
5. The calculator can memorize 1 number
6. The calculator can retrieve the memorized number
7. Implement the multiplication operator function for integers
8. Implement the division operator for positive integers
9. Implement the modular operator function
10. Implement the absolute function
11. Testing all the above functions with at least five different cases
12. Raising relevant exceptions when needed
13. Including relevant comments

**Milestone #2-November 20th 11:59pm**

1. Change user input to handle as many calculation as needed [/10]
   1. Ignore BEDMAS
2. Change user input to handle decimal values as well [/10]
   1. It will always be in the form
   2. #.# : always a number before/after the decimal
      1. You won’t see 1230.
      2. You won’t see ,0123
      3. You always something 12.12
3. Implement the multiplication operator for floats [/5]
4. Implement the division operator for both positive and negative **floats** [/10]
5. Implementing factorial [/5]
   1. **Recursive method is not allowed**
6. Can memorize and retrieve unlimited numbers [/10]
7. Create relevant functions where needed
8. Testing all the above functions with at least three different cases
   1. Test using floats for multi/divide
9. Raising relevant exceptions when needed
10. Including relevant comments

Hint #1:

You will need the .shift() method so check it out here

<https://www.geeksforgeeks.org/python-decimal-shift-method/>

Hint #2 : Handle as many calculation as needed

|  |
| --- |
| “1+2+3+4” → “3+3+4” → “6+4” → “10”  “4\*4\*4\*4” → “16\*4\*4” → “64\*4” → “256”  “1+2+3+” → “3+3+4” → “6+” → Should raise an Exception  This step-by-step conversion will definitely need a while loop   * Retrieve the first number it sees, * Retrieve the first operator it sees * Retrieve the second number it sees * Replace it with the result. Then repeat. |

You might find the following methods to be helpful:

<https://www.w3schools.com/python/python_ref_string.asp>

* .replace()
* .isdigit() or .isnumeric() or .isdecimal()

Hint #3: Absolute value

* Before you perform any calculations, it might be best to handle all the absolute firsts

|  |
| --- |
| “1+|-2|+3+|-4|” → “1+2+3+|-4|” → “1+2+3+4” → “3+3+4” → “6+4” → “10” |

Hint #4 : Handle memorize/retrieve unlimited numbers

* Look at milestone 1 solutions, see how one memory is done.
* Create two Lists
  + Think about it. Why specifically two Lists? Why not just one?
* There’s something you can use from this <https://www.w3schools.com/python/python_ref_list.asp>

Hint #5 :

* For factorial, call the multiply function

Hint #6 :

* For handling multiplication and division of floats, do not change multiply and divide functions. Instead create two more functions that calls multiply and divide
  + In these two new functions, use shift somehow
  + use a while loop to figure out how many shifts you need for both values and then unshift the result at the end
    - Example
    - 12.1 \* 0.5 → 121 \* 5 = 605 → 6.05
      * Two shifts to the RIGHT
      * Two shifts to the left
    - multi : product = float(product) \* 10 \*\* - shifts
    - divide : quotient = float(quotient) \* 10 \*\* - shifts

**Milestone #3-November 27th 11:59pm**

1. Handle BEDMAS [/10]
2. Implement the power operator where the power are only positive/negative integers [/10]
3. Implementing reciprocal [/5]
4. Implementing the square root function[/5]
5. Implementing a function to calculate the exact value of pi to 17 decimal values [/5]
6. Implementing a function to calculate the value of e to 17 decimal values [/5]
7. Create relevant functions where needed
8. Raising relevant exceptions when needed
9. Including relevant comments
10. Testing all the above functions with at least three different cases

**Milestone #4-December 4th 11:59pm**

1. Implementing brackets
2. Implementing power operator with floats
3. Implementing log or ln
4. Implementing sine/cosine/tan
5. Implementing inverse sine/cosine/tan
6. Create relevant functions where needed
7. Testing all the above functions with at least three different cases
8. Raising relevant exceptions when needed
9. Including relevant comments

**Participation Marks (/25 marks each milestone)**

* 5 mark for each day of contributions
* It is expected that you will use class time properly.
  + Ask for help
  + I will not be giving you the answer
* **At the end of each class, you have to submit something and I have to see that it is significantly different from the previous class’s submission.**

**Testing/Exception Marks (/15 marks each milestone)**

* Creating at least three test cases for /10 of the marks
* Getting /5 marks for raising at least two Exceptions in an “appropriate” place

**Comments Marks (/10 marks each milestone)**

* Milestone #1
  + 2 marks for your name somewhere
  + 1 mark for multi
  + 1 mark for division
  + 1 mark for modular
  + 1 mark for absolute
  + 4 marks for your user input handling
* https://www.w3schools.com/python/python\_comments.asp
* Milestone #2
  + Writing at least some lines of comments to describe your code
    - 2 marks for factorial
    - 1 marks for changing multi
    - 1 marks for changing divide
    - 4 marks for both memory functions
    - 1 marks for decimal handling
    - 1 marks for as many calculation as needed

**Restrictions**

* You are not allowed to use the following operators inside the functions
  + Multiplication : \*
  + Division : /
  + Modular : %
  + Power : \*\*
* You are not allowed to use any of the following functions/code
  + abs()
  + import math
  + eval()

**FAQ In General**

* Can we use the restricted operators to test our code?
  + Of course!
* What happens if I can’t get certain functionalities?
  + You will still get part marks for writing test cases for them or raising a relevant Exception
  + You will also get marks for writing comments
* Can we change how we handle user input
  + Yes...however you must still implement all the functionality
    - Doing all the operators
    - Doing memory
    - Calculating one pairs of numbers with an operator
  + You have to call the functions you need to create when doing the mathematical operators
    - Multi
    - Divide
    - Absolute
    - Subtract
    - Add
    - Modular
  + You have to test the functions using assert() properly
    - Don’t need to test using assert with memory
* What if I don’t know what I am doing at all?
  + At least write the function definition inputs,
    - You can return 0 to start for each of them
  + Write the test cases for the function definitions
  + Write two relevant exceptions
  + Add some relevant comments
  + This should guarantee you to get at least 50% on each milestone

**FAQ For Milestone 2**

* Is it only floats for the user input?
  + The user should be able to enter both integers and floats
* Will an operation be inside absolute value?
  + No, this would be too hard to do for milestone two: you won’t have to handle |1+1|
  + Absolute values will only surround a number

**Marking Rubric**

* Calling functions
* Inputs
* Strings
* Lists
* If statements
* Iteration : For using values
* Iteration : For using range/len
* Repetition : while
* Continue/Break
* Creating Functions
* Global and Local Variables
* Testing with Assert
* Errors
* Exceptions
* Debugger

**Opportunity to replace a zero on a homework**

* If you have previously received any zeros on a homework [Homework 1-20], you can ask for an opportunity to regain that lost mark.
* To do so, ask me for an opportunity to replace them.
  + The opportunity would be to implement something “extra” for your milestones

**Bonus Mark Opportunities**

* Anything extra you want to add in
  + Using images
  + Using audio
  + Saving it somewhere
* Ask me first if you want to add something for “bonus” marks