CIS 61 :: Lab 01 - Expressions and Functions

Student Name:

Instructions:

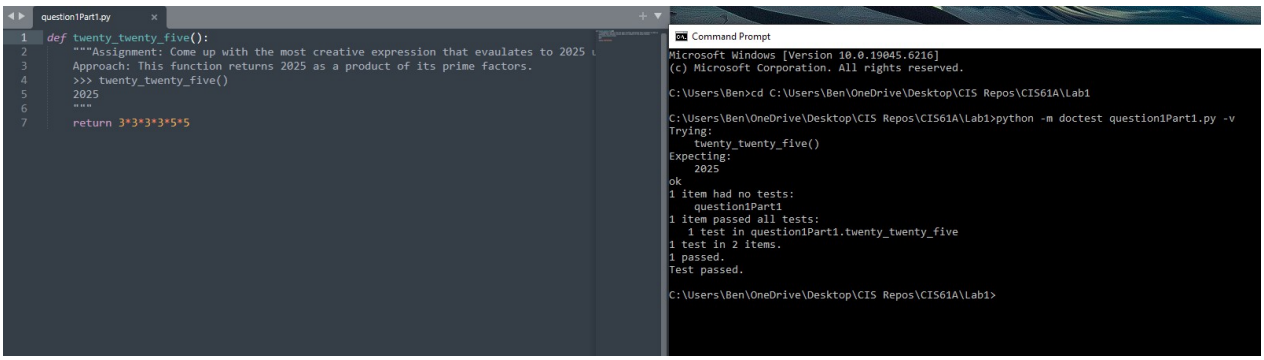
1. *Make a copy of the assignment template. Go to File => Make a copy (or download as a Word file).*
2. Attach Snipping Photos for each question.
3. *Place your name in the Title of each Assignment*
   a. *For Example: CIS 61 - Lab 01 - Expression and Names - Irfan O***.**
4. **Submission:** When done, go to **File -> Download as -> Microsoft Word** and then upload the file to Canvas.

Lab 1 - Expressions and Functions

Instructions: Use Sublime text editor to write your code and use Python shell to execute the below programs. Attach Snipping photos of **your source code** and **executions of the code in Python shell**.

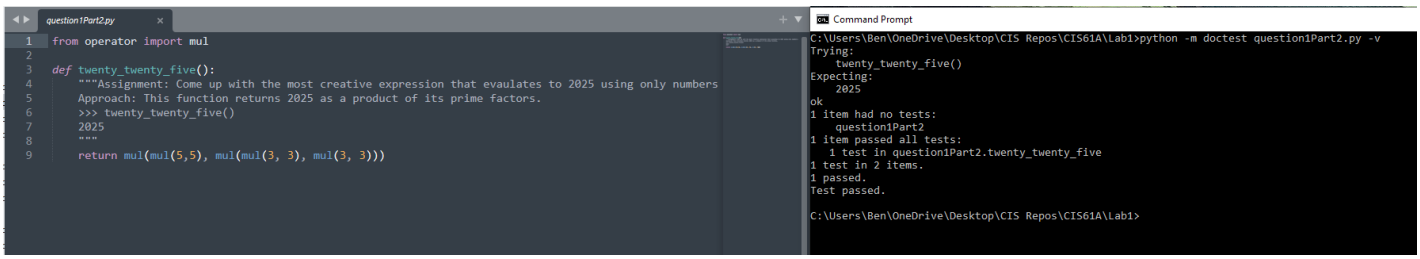
Question 1: Twenty-Twenty-Five

Part 1: Come up with the most creative expression that evaluates to 2025, using only numbers and the [+, *, -] operators. Use integers. Do not use the round function..



Part 2: Try to rewrite the same expression, this time entirely with call expressions. You can just use the terminal as well.

(using function calls: add, mul, sub, etc… You can use **from operators import add, mul, sub**)



Question 2: Area

Part 1: Write definitions for these functions:

rect_area(width, height) returns the area of a rectangle giving its dimensions.

rect_peri (width, height) returns the perimeter of a rectangle giving its dimensions.

question2Part1.py

```python
def rect_area(width, height):
    """ Area of a rectangle
    >>> rect_area(5, 10)
    50
    """
    return width*height

def rect_perimeter(width, height):
    """ Perimeter of a rectangle
    >>> rect_perimeter(5, 10)
    30
    """
    return 2*(width+height)
```

Command Prompt

```
C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>python -m doctest question2Part1.py -v
Trying:
    rect_area(5, 10)
Expecting:
    50
ok
Trying:
    rect_perimeter(5, 10)
Expecting:
    30
ok
1 item had no tests:
    question2Part1
2 items passed all tests:
   1 test in question2Part1.rect_area
   1 test in question2Part1.rect_perimeter
2 tests in 3 items.
2 passed.
Test passed.

C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>
```

Part 2: Rewrite the above functions with lambda expressions and assign them to respective names. You can just use a Python shell and take the screenshot of the code.

```python
rect_area = lambda width, height: width*height

rect_perimeter = lambda width, height: 2(width+height)
```

Question 3: Rain or Shine

Part 1: Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining. Write a function that takes in the current temperature and a boolean value telling if it is raining and it should return True if Alfonso will wear a jacket and False otherwise.

Try solving this problem with a single line of code.

question2Part1.py | question3Part1.py

```python
def wears_jacket(temp, raining):
    """
    >>> wears_jacket(90, False)
    False
    >>> wears_jacket(40, False)
    True
    >>> wears_jacket(100, True)
    True
    """
    return temp < 60 or raining
```

Select Command Prompt

```
C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>python -m doctest question3Part1.py -v
Trying:
    wears_jacket(90, False)
Expecting:
    False
ok
Trying:
    wears_jacket(40, False)
Expecting:
    True
ok
Trying:
    wears_jacket(100, True)
Expecting:
    True
ok
1 item had no tests:
    question3Part1
1 item passed all tests:
   3 tests in question3Part1.wears_jacket
3 tests in 2 items.
3 passed.
Test passed.

C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>
```

*Note that it should either return **True** or **False** based on a single condition, whose truthiness value will also be either True or False.*

Part 2: Rewrite the above function with a lambda expression. You can just use a Python shell and take a screenshot of the code.

question2Part1.py | question3Part2.py

```python
wears_jacket = lambda temp, raining: temp < 60 or raining
```

Question 4: Sum of the first N natural numbers:

Part 1: Write a function sumNaturals (n) that returns the sum of the first n natural numbers. You can use this formula 1 + 2 + ... + n = n(n+1) / 2.
Make sure that the function returns an integer.

Do not use a for loop or a while loop.

question4Part1.py

```python
def sumNaturals(n):
    """ Sum all the first n natural numbers.
    >>> sumNaturals(3) # 1 + 2 + 3 = 6
    6
    >>> sumNaturals(5) # 1 + 2 + 3 + 4 + 5 = 15
    15
    """
    return n*(n+1)//2
```

Command Prompt

```
C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>python -m doctest question4Part1.py -v
Trying:
    sumNaturals(3) # 1 + 2 + 3 = 6
Expecting:
    6
ok
Trying:
    sumNaturals(5) # 1 + 2 + 3 + 4 + 5 = 15
Expecting:
    15
ok
1 item had no tests:
    question4Part1
1 item passed all tests:
   2 tests in question4Part1.sumNaturals
2 tests in 2 items.
2 passed.
Test passed.

C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>
```

Part 2: Define a lambda expression that takes **n** and returns the sum of the first **n** natural numbers, using the above formula. You can just use a Python shell and take a screenshot of the code.

question4Part2.py

```
sumNaturals = lambda n: n*(n+1)//2
```

Q5: A Plus Abs B

question5.py

```python
from operator import add, sub

def a_plus_abs_b(a, b):
    """Return a+abs(b), but without calling abs.
    >>> a_plus_abs_b(2, 3)
    5
    >>> a_plus_abs_b(2, -3)
    5
    """

    if b < 0:
        f = sub
    else:
        f = add

    return f(a, b)
```

Command Prompt

```
C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>python -m doctest question5.py -v
Trying:
    a_plus_abs_b(2, 3)
Expecting:
    5
ok
Trying:
    a_plus_abs_b(2, -3)
Expecting:
    5
ok
1 item had no tests:
    question5
1 item passed all tests:
   2 tests in question5.a_plus_abs_b
2 tests in 2 items.
2 passed.
Test passed.

C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>
```

Question 6: You Define a Function

Part 1: Write a function that takes in one or two input parameters and returns an output. The function should return the output of a **one-line expression**. Write at least three test cases for your function in the docstring. Use the command line to test your function against the test cases. Take a screenshot of your code and the result of your test. Also, write the function in the below box as well.

Make sure your function has just one line of code

question6Part1.py

```python
from operator import mul, sub

def multiplyButFrustrating (variableOne, variableTwo):
    """Multiplies variableOne by variableTwo
    >>> multiplyButFrustrating(2,6)
    11
    >>> multiplyButFrustrating(0,3)
    -1
    >>> multiplyButFrustrating(5,-1)
    -6
    """
    return sub(mul(variableOne, variableTwo), 1)
```

Command Prompt

```
C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>python -m doctest question6Part1.py -v
Trying:
    multiplyButFrustrating(2,6)
Expecting:
    11
ok
Trying:
    multiplyButFrustrating(0,3)
Expecting:
    -1
ok
Trying:
    multiplyButFrustrating(5,-1)
Expecting:
    -6
ok
1 item had no tests:
    question6Part1
1 item passed all tests:
   3 tests in question6Part1.multiplyButFrustrating
3 tests in 2 items.
3 passed.
Test passed.

C:\Users\Ben\OneDrive\Desktop\CIS Repos\CIS61A\Lab1>
```

Part 2: Write the same function as a lambda function.

question6Part1.py question6Part2.py

```python
from operator import mul, sub

multiplyButFrustrating = lambda variableOne, variableTwo: sub(mul(variableOne, variableTwo), 1)
```