# **CIS 61 :: Lab 01 - Expressions and Names**

#### **Student Name: Jonathan Adrianto Saleh (CCCID : CAU9032)**

|  |
| --- |
| **Instructions:**  1. *Make a copy of the assignment template. Go to File => Make a copy (or download as a Word file.)* 2. Complete definitions and attach Snipping Photos where appropriate 3. *Place your name in the Title of each Assignment*    1. *For Example: CIS 61 - Lab 00 - Introduction to Python - Irfan O***.** 4. Use the book or do online research to find answers. 5. **Write your answers using a different font color. Find your own unique color.** 6. Write answers in your own words. **DO NOT COPY & PASTE** from anywhere. 7. **Submission:** When done, go to **File -> Download as -> Microsoft Word** and then upload the file to Lab-00 on Canvas. |

*The lines in the triple-quotes """ are called a* ***docstring****, which is a description of what the function is supposed to do. When writing code in 61A, you should always read the docstring!*

*The lines that begin with >>> are called* ***doctests****. Recall that when using the Python interpreter, you write Python expressions next to >>> and the output is printed below that line. Doctests explain what the function does by showing actual Python code: "if we input this Python code, what should the expected output be?"*

*Don't forget to save your assignment after you edit it! In most text editors, you can save by navigating to File > Save or by pressing Command-S on MacOS or Ctrl-S on Windows.*

Below command will help you test the python file

**> python3 -m doctest -v filename.py**

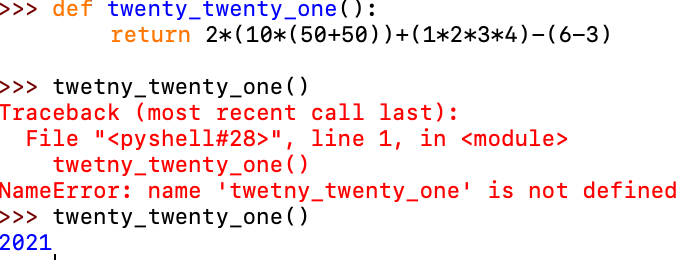
**Lab 1 - Expressions and Functions – Jonathan Saleh**

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

**Question 1**: **Twenty-Twenty-One**

**Part 1:** Come up with the most creative expression that evaluates to 2021, using only numbers and the +, \*, and - operators.   
You should replace the underscores in return \_\_\_\_\_\_ with the expression that evaluates to 2020.

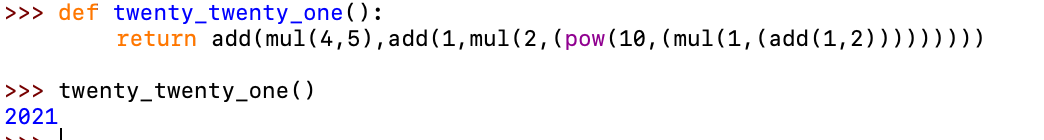
|  |
| --- |
| **def** **twenty\_twenty\_one**():  """Come up with the most creative expression that evaluates   to 2021,using only numbers and the +, \*, and - operators.  (no call expressions)  >>> twenty\_twenty\_one()  2021  """  **return** 2\*(10\*(50+50))+(1\*2\*3\*4)-(6-3) |



**Part 2:** Try to rewrite the same expression, this time entirely with call expressions (using function calls: add, mul, pow, etc..)

**>>> def twenty\_twenty\_one():**

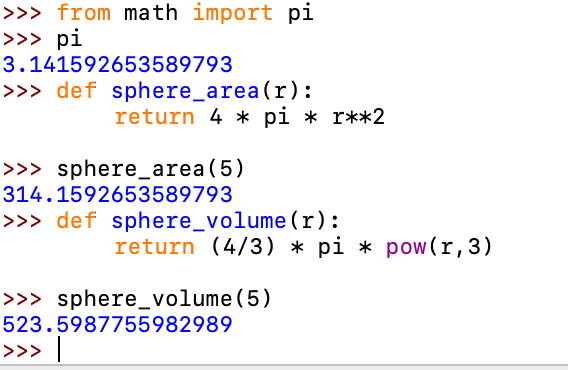
**return add(mul(4,5),add(1,mul(2,(pow(10,(mul(1,(add(1,2)))))))))**



**Question 2**: **Area**  
Part 1: Write definitions for these functions:   
sphereArea(radius) returns the surface area of a sphere having the given radius.   
sphere Volume (radius). Returns the volume of a sphere having the given radius.

Write two test cases for each function. Be careful! These functions will return decimal numbers (floats).

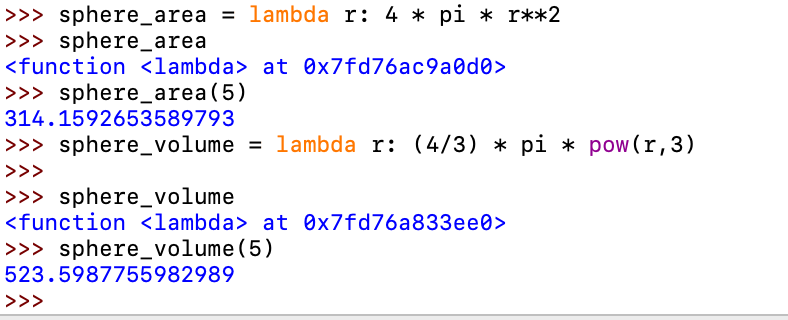
|  |
| --- |
| **from math import pi**  **def** **sphere\_area**(r):  """Area of a sphere with radius r.  >>> sphere\_area(5)  314.14159  >>> sphere\_area(10)  1256.6370  """  return 4\* pi \* r\*\*2  **def** **sphere\_volume**(r):  """Volume of a sphere with radius r.  >>> sphere\_area(5)  523.59877  >>> sphere\_area(10)  4188.79020  """  Return (4/3) \* pi \* pow(r,3) |

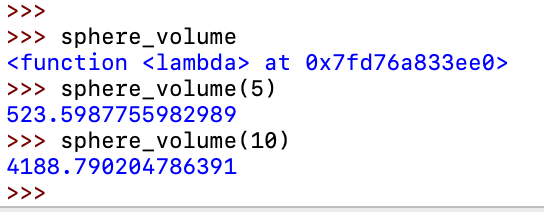
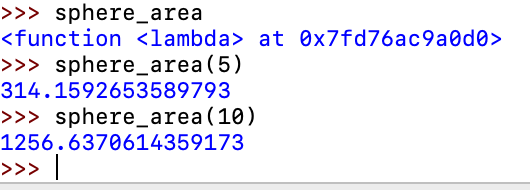


Part 2: Rewrite above functions with lambda expressions and assign them to respective names. You can just use a Python shell and take the screenshot of the code.

**sphere\_area** = lambda r: 4\* 3.14 \* r\*\*(2)

**sphere\_volume** = lambda r:(4/3)\*3.14\* pow(r,3)





**Question 3: Rain or Shine**

Part 1: Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining.

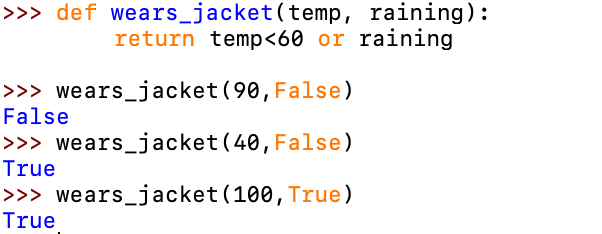
Write a function that takes in the current temperature and a boolean value telling

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.**

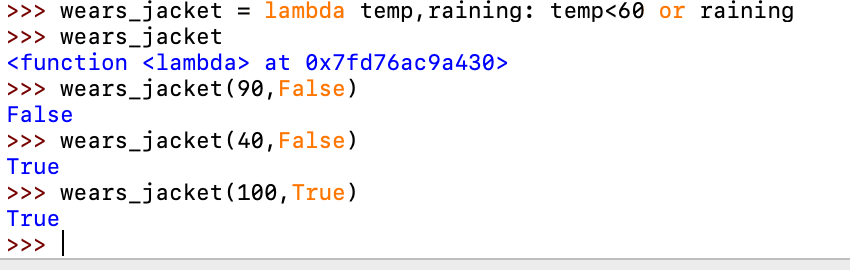
|  |
| --- |
| **def wears\_jacket(**temp, raining**):**  """  >>> wears\_jacket(90, False)  False  >>> wears\_jacket(40, False)  True  >>> wears\_jacket(100, True)  True  """  return temp<60 or raining |

*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 above function with a lambda expression. You can just use a Python shell and take the screenshot of the code.

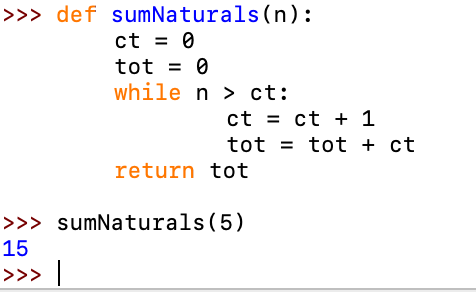
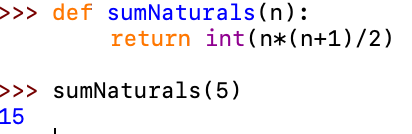
**wears\_jacket = lambda temp,raning: temp<60 or raining**



**Question 4: Sum of the first N natural numbers**:

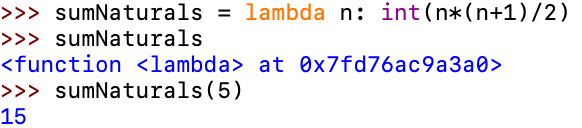
**Part 1:** Write a function sumNaturals (n) that returns the sum of the first n natural numbers. You can use this formula 1+ 2+ ... + n = n(n+1) / 2 or you can use a **while loop** with a count and an accumulator variable. Make sure that the function returns an int. **Do not use a for loop**.

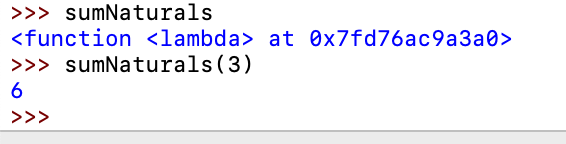
|  |
| --- |
| **def** **sumNaturals**(n):  """Sum all the first n natural numbers.  >>> sumNaturals(3) # 1 + 2 + 3 = 6  6  >>> sumNaturasl(5) # 1 + 2 + 3 + 4 + 5 = 15  15  """  ct = 0  tot = 0  while n > ct:  ct = ct + 1  tot = tot + ct  return tot |

**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 the screenshot of the code.

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



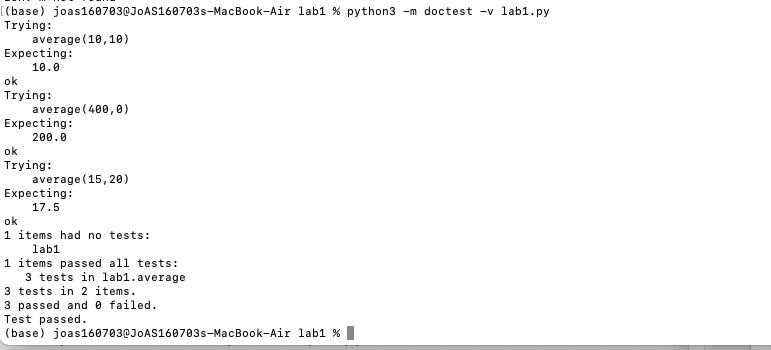


**Question 5: You Define a Function**

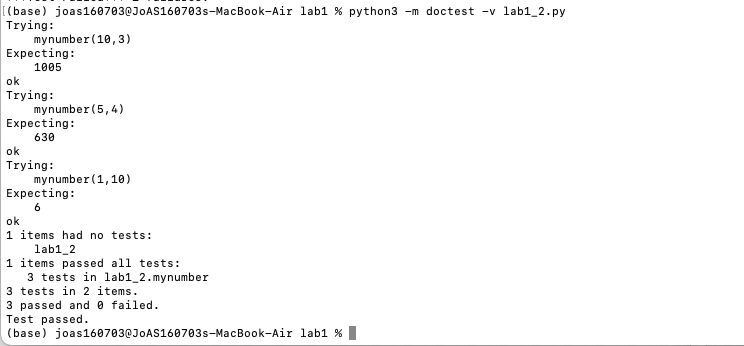
Part 1: Write a function that takes in one or two inputs and returns an output. The function should return the output of a one-line expression. Write at least three test cases for your function in the docstring. Use the command line to test your function against the test cases. Take a screenshot of your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

|  |
| --- |
| **def** **average**(n1,n2):  """Average of two Numbers  >>> average(10,10)  10.0  >>> average(400,0)  200.0  >>> average(15,20)  17.5  """  return (n1 + n2)/2 |
| **def** **mynumber**(x,y):  """makes x the power of y and adds 5  >>> mynumber(10,3)  1005  >>> mynumber(5,4)  630  >>> mynumber(1,10)  6  """  return add(5,pow(x,y)) |
| **def** **buy\_food**(hungry,money):  """Should buy food  >>> **buy\_food** (True,10)  False  >>> **buy\_food** (False,100)  False  >>> **buy\_food** (True,20)  True  """  return hungry and money>10 |

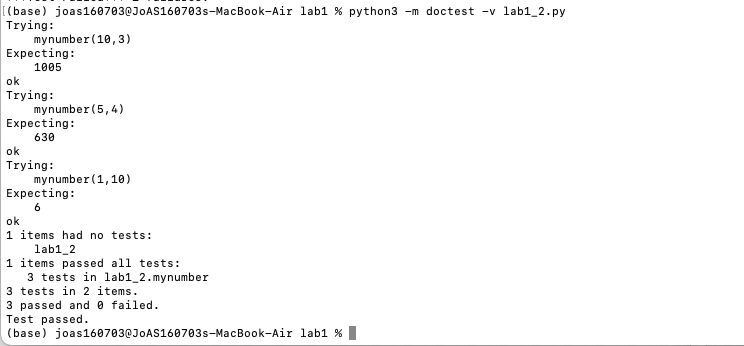
Program 1



Program 2

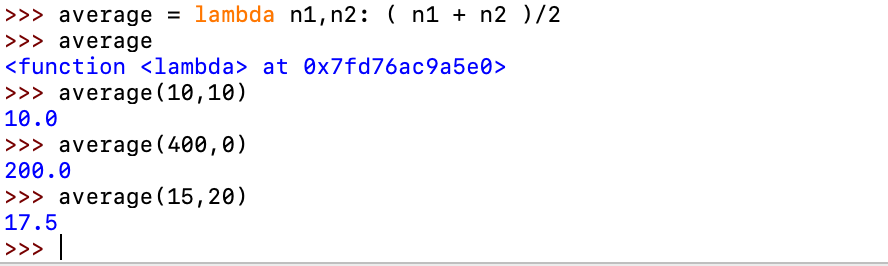


Program 3

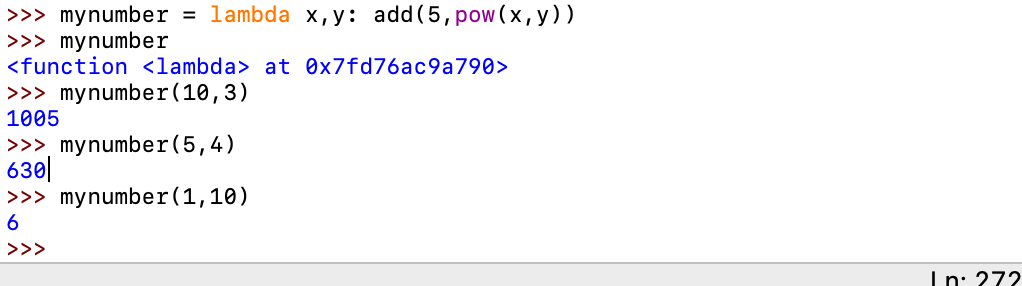


Part 2: Write the same function as a lambda function.

average = lambda n1,n2: ( n1 + n2 )/2



mynumber = lambda x,y: add(5,pow(x,y))



buy\_food = lambda hungry,money: hungry and money>10

