PSTAT 5A Lab 1 - SOLUTIONS

Introduction to Python and JupyterHub

Solution Key

2025-07-29

Table of contents



# Lab 1 Solutions

This document provides complete solutions to all tasks in PSTAT 5A Lab 1.

## Task 1 Solution

**Objective:** Rename your notebook from Untitled to Lab1

**Steps:**

* Located the notebook in the file browser (appears as Untitled.ipynb)
* Right-clicked on the notebook name
* Selected “Rename” from the context menu
* Changed the name to Lab1
* Pressed Enter to confirm
* Verified the title bar now shows Lab1.ipynb

**Expected Result:**

Your notebook should now be named Lab1.ipynb and this should be visible in both the file browser and the title bar.

## Task 2 Solution

**Objective:** Create a Markdown cell with heading and a Code cell with basic arithmetic

**Markdown Cell:**

# Task 2

**Code Cell:**

2 + 2

4

**Expected Result:**

* The code cell executes the arithmetic operation
* Python displays the result 4 below the cell
* A new empty code cell automatically appears below
* The cell is marked as executed with a number like [1]

## Task 3 Solution

**Objective:** Demonstrate syntax error and understand error messages

**Markdown Cell:**

# Task 3

**Code Cell with Intentional Error:**

2 plus 2

**Expected Error Output:**

Cell In[2], line 1  
 2 plus 2  
 ^^^^  
SyntaxError: invalid syntax

**Explanation:**

* Python doesn’t understand plus as an operator
* The ^^^^ points to where Python detected the problem
* The error message tells us it’s a SyntaxError meaning invalid Python syntax
* In Python, we must use + for addition, not the word plus

**Corrected Version:**

2 + 2 # This works correctly

4

## Task 4 Solution

**Objective:** Compute mathematical expressions using Python\*\*

**Problem 1:**

**Python Code:**

(2 + 3) / (4 + 5\*\*6)

0.0003199181009661527

**Step by Step:**

numerator = 2 + 3  
print(f"Numerator: {numerator}")  
  
denominator = 4 + 5\*\*6  
print(f"Denominator: {denominator}")  
  
result = numerator / denominator  
print(f"Final result: {result}")

Numerator: 5  
Denominator: 15629  
Final result: 0.0003199181009661527

**Problem 2:**

**Python Code:**

(1 - 3 \* 4\*\*5)\*\*6

838839550121163601921

**Step by Step:**

inner\_exponent = 4\*\*5  
print(f"4^5 = {inner\_exponent}")  
  
multiplication = 3 \* inner\_exponent  
print(f"3 \* 4^5 = {multiplication}")  
  
subtraction = 1 - multiplication  
print(f"1 - 3 \* 4^5 = {subtraction}")  
  
final\_result = subtraction\*\*6  
print(f"(1 - 3 \* 4^5)^6 = {final\_result}")

4^5 = 1024  
3 \* 4^5 = 3072  
1 - 3 \* 4^5 = -3071  
(1 - 3 \* 4^5)^6 = 838839550121163601921

## Task 5 Solution

**Objective:** Understand module importing and fix NameError

**Step 1: Code that produces error**

sin(1)

**Expected Error:**

NameError: name 'sin' is not defined

**Explanation:** Python doesn’t recognize sin because the math functions aren’t loaded by default.

**Step 2: Import module and retry**

from math import \*  
sin(1)

0.8414709848078965

**Alternative Solutions:**

# Method 1: Import specific function  
from math import sin  
print(sin(1))  
  
# Method 2: Import entire module  
import math  
print(math.sin(1))  
  
# Method 3: Import with alias  
import math as m  
print(m.sin(1))

0.8414709848078965  
0.8414709848078965  
0.8414709848078965

All produce the same result: 0.8414709848078965

## Task 6 Solution

**Objective:** Understand Python case sensitivity

**Step 1: Variable assignment**

my\_variable = 5

**Step 2: Wrong capitalization**

print(My\_variable)

**Expected Error:**

NameError: name 'My\_variable' is not defined

**Explanation:** Python is case-sensitive, so My\_variable ≠ my\_variable

**Step 3: Correct capitalization**

print(my\_variable)

5

**Additional Examples:**

# These are all different variables in Python  
my\_variable = 5  
My\_variable = 10  
MY\_VARIABLE = 15  
my\_Variable = 20  
  
print(f"my\_variable = {my\_variable}")  
print(f"My\_variable = {My\_variable}")  
print(f"MY\_VARIABLE = {MY\_VARIABLE}")  
print(f"my\_Variable = {my\_Variable}")

my\_variable = 5  
My\_variable = 10  
MY\_VARIABLE = 15  
my\_Variable = 20

## Task 7 Solution

**Objective:** Add descriptive comments to previous code

**Examples of well-commented code from previous tasks:**

# Task 2: Basic arithmetic  
2 + 2 # Adding two integers  
  
# Task 4: Complex mathematical expression  
# Calculate (2 + 3) / (4 + 5^6)  
numerator = 2 + 3 # Sum of 2 and 3  
denominator = 4 + 5\*\*6 # 4 plus 5 to the 6th power  
result = numerator / denominator # Final division  
print(f"Result: {result}")  
  
# Task 5: Import math module and use sin function  
from math import \* # Import all math functions  
angle\_in\_radians = 1 # Input angle in radians  
sine\_value = sin(angle\_in\_radians) # Calculate sine  
print(f"sin(1) = {sine\_value}")  
  
# Task 6: Variable assignment with proper naming  
my\_variable = 5 # Store the value 5 in my\_variable  
print(my\_variable) # Display the value  
  
"""  
This is a multi-line comment.  
It can span multiple lines and is useful  
for longer explanations or documentation.  
"""

**Good commenting practices demonstrated:**

* Explain what the code does
* Clarify complex calculations
* Document variable purposes
* Use both inline (#) and block (""") comments

## Task 8 Solution

**Objective:** Explore Python data types using the type() function

**Code and Expected Outputs:**

print(type(1)) # Output: <class 'int'>  
print(type(1.1)) # Output: <class 'float'>  
print(type("hello")) # Output: <class 'str'>

<class 'int'>  
<class 'float'>  
<class 'str'>

**Additional Examples:**

# More data type examples  
print("Integer:", type(42))  
print("Float:", type(3.14159))  
print("String with single quotes:", type('Python'))  
print("String with double quotes:", type("Programming"))  
print("Boolean True:", type(True))  
print("Boolean False:", type(False))  
print("List:", type([1, 2, 3]))  
print("Tuple:", type((1, 2, 3)))  
print("Dictionary:", type({"key": "value"}))

Integer: <class 'int'>  
Float: <class 'float'>  
String with single quotes: <class 'str'>  
String with double quotes: <class 'str'>  
Boolean True: <class 'bool'>  
Boolean False: <class 'bool'>  
List: <class 'list'>  
Tuple: <class 'tuple'>  
Dictionary: <class 'dict'>

## Task 9 Solution

**Objective:** Practice variable assignment, updating, and calculations

**Markdown cell:**

# Task 9

**Step 2: Initial variable assignments**

course = "PSTAT 5A"  
num\_sections = 4  
section\_capacity = 25

**Step 3: Update num\_sections (correct approach)**

num\_sections = num\_sections + 1  
print(f"Updated number of sections: {num\_sections}")  
# Alternative: num\_sections += 1  
# Alternative: num\_sections = 4 + 1

Updated number of sections: 5

**Step 4: Predict and test expressions**

print(type(course)) # Expected: <class 'str'>  
print(type(num\_sections)) # Expected: <class 'int'>  
print(num\_sections \* section\_capacity) # Expected: 125

<class 'str'>  
<class 'int'>  
125

**Step 5: Calculate course capacity**

course\_capacity = num\_sections \* section\_capacity  
print(f"Course: {course}")  
print(f"Number of sections: {num\_sections}")  
print(f"Capacity per section: {section\_capacity}")  
print(f"Total course capacity: {course\_capacity}")

Course: PSTAT 5A  
Number of sections: 5  
Capacity per section: 25  
Total course capacity: 125

**Complete Solution with Comments:**

# Step 2: Initial variable assignments  
course = "PSTAT 5A" # Course name as string  
num\_sections = 4 # Initial number of sections  
section\_capacity = 25 # Maximum students per section  
  
# Step 3: A new section has been added  
num\_sections = num\_sections + 1 # Increment by 1, now equals 5  
  
# Step 4: Testing expressions with predictions  
print("Testing type() function:")  
print(f"type(course) = {type(course)}") # Expected: <class 'str'>  
print(f"type(num\_sections) = {type(num\_sections)}") # Expected: <class 'int'>  
print(f"num\_sections \* section\_capacity = {num\_sections \* section\_capacity}") # Expected: 125  
  
# Step 5: Calculate total course capacity  
course\_capacity = num\_sections \* section\_capacity # 5 × 25 = 125  
print(f"\nFinal Results:")  
print(f"Course: {course}")  
print(f"Total sections: {num\_sections}")  
print(f"Capacity per section: {section\_capacity}")  
print(f"Total course capacity: {course\_capacity} students")

Testing type() function:  
type(course) = <class 'str'>  
type(num\_sections) = <class 'int'>  
num\_sections \* section\_capacity = 125  
  
Final Results:  
Course: PSTAT 5A  
Total sections: 5  
Capacity per section: 25  
Total course capacity: 125 students

## Summary of Key Concepts Learned

**✅ JupyterHub Environment**

* Creating and renaming notebooks
* Understanding cell types (Code vs Markdown)
* Running cells with ▶️ button or Shift+Enter
* Navigating the interface

**✅ Python Basics**

* Arithmetic operations: +, -, \*, /, \*\*
* Order of operations: Parentheses, Exponents, Multiplication/Division, Addition/Subtraction
* Error reading: Understanding SyntaxError and NameError messages

**✅ Variables and Data Types**

* Variable assignment: variable\_name = value
* Case sensitivity: my\_var ≠ My\_var
* Basic types: int, float, str, bool
* Type checking: type() function

**✅ Modules and Imports**

* Import syntax: from module import \* or import module
* Using functions: After importing, functions become available
* Math module: Contains mathematical functions like sin(), cos(), etc.

**✅ Comments and Documentation**

* Inline comments: # This is a comment
* Block comments: """Multi-line comment"""
* Purpose: Document code for yourself and others

**✅ Programming Best Practices**

* Write descriptive variable names
* Add comments to explain complex logic
* Test your code incrementally
* Read and understand error messages
* Use existing variables in calculations when possible

## Next Steps

In Lab 2, you’ll learn about:

* Python functions and how to create them
* Data structures (lists, dictionaries)
* Control flow (if statements, loops)
* More advanced programming concepts

**Great work completing Lab 1!** You now have the foundation needed for statistical programming in Python.