Python Lab 1 Exercises – Strings and Numbers

Part I: String Manipulation

(source: https://www.pythonforbeginners.com/basics/string-manipulation-in-python)

Table of Contents

What's a String in Python?

A python string is a list of characters in an order. A character is anything you can type on the keyboard in one keystroke, like a letter, a number, or a backslash. Strings can also have spaces, tabs, and newline characters. For instance, given below is a string in Python.

```
>>> myStr="hello world"
```

Here, we have assigned the string "hello world" to the variable myStr. We can also define an empty string that has 0 characters as shown below.

```
>>> myStr=""
```

In python, every string starts with and ends with quotation marks i.e. single quotes ' ', double quotes " " or triple quotes """ """.

String Manipulation in Python

String manipulation in python is the act of modifying a string or creating a new string by making changes to existing strings. To manipulate strings, we can use some of Pythons built-in methods.

Create a String in Python

To create a string with given characters, you can assign the characters to a variable after enclosing them in double quotes or single quotes as shown below.

```
>>> word = "Hello World"
>>> print(word)
```

Output:

```
Hello World
```

In the above example, we created a variable word and assigned the string "Hello World" to the variable. You can observe that the string is printed without the quotation marks when we print it using the **print()** function.

Access Characters in a String in Python

To access characters of a string, we can use the python indexing operator [ ] i.e. square brackets to access characters in a string as shown below.

```
>>> word = "Hello World"
```

```
>>> print("The word is:",word)
>>> letter=word[0]
>>> print("The letter is:",letter)
```

Output:

```
The word is: Hello World
The letter is: H
```

In the above example, we created a string “Hello World” and assigned it to the variable word. Then, we used the string indexing operator to access the first character of the string and assigned it to the variable letter.

Find Length of a String in Python

To find the length of a string in Python, we can use the **len()** function. The **len()** function takes a string as input argument and returns the length of the string as shown below.

```
word = "Hello World"
print("The string is:",word)
length=len(word)
print("The length of the string is:",length)
```

Output:

```
The string is: Hello World
The length of the string is: 11
```

In the above example, the **len()** function returns 11 as the length of the string. This is due to the reason that there are 11 characters in the string including the space character.

Find a Character in a String in Python

To find the index of a character in a string, we can use the **find()** method. The **find()** method, when invoked on a string, takes the character as its input argument and returns the index of first occurrence of the character as shown below.

```
>>> word = "Hello World"
>>> print("The string is:",word)
>>> character="W"
>>> print("The character is:",character)
>>> position=word.find(character)
>>> print("The position of the character in the string is:",position)
```

Output:

```
The string is: Hello World
The character is: W
The position of the character in the string is: 6
```

In this example, we searched for the character "W" in the string using the find() method. As "W" is present at index 6 of the string, the **find()** method gives the value 6 as its output.

If the character is not present in the string, the **find()** method gives -1 as its output. You can observe this in the following example.

```
>>> word = "Hello World"
>>> print("The string is:",word)
>>> character="Z"
>>> print("The character is:",character)
>>> position=word.find(character)
>>> print("The position of the character in the string is:",position)
```

Output:

```
The string is: Hello World
The character is: Z
The position of the character in the string is: -1
```

In this example, we passed the character "Z" to the **find()** method as its input. You can observe that the **find()** method returns -1 as there is no "Z" in the string "Hello World".

You can also find the index of a character or a substring in a string using the **index()** method. The **index()** method, when invoked on a string, takes a character or substring as its input argument. After execution, it returns the index of first occurrence of character or substring as shown below.

```
>>> word = "Hello World"
>>> print("The string is:",word)
>>> character="W"
>>> print("The character is:",character)
>>> position=word.index(character)
>>> print("The index of the character in the string is:",position)
```

Output:

```
The string is: Hello World
The character is: W
The index of the character in the string is: 6
```

Find Frequency of a Character in a String in Python

You can also perform string manipulation in python to find the frequency of a character in the string. For this, we can use the **count()** method. The **count()** method, when invoked on a string, takes a character as its input argument and returns the frequency of the character as shown below.

```
>>> word = "Hello World"
>>> print("The string is:",word)
>>> character="l"
>>> print("The character is:",character)
>>> position=word.count(character)
>>> print("The frequency of the character in the string is:",position)
```

Output:

```
The string is: Hello World
```

The character is: l

The frequency of the character in the string is: 3

In the above example, we used the **count()** method find the frequency of the character "l" in the string "Hello World". As there are three instances of "l" in "Hello World", the **count()** method gives 3 as its output.

Count the Number of Spaces in a String in Python

Spaces are also characters. Hence, you can use the **count()** method count the number of spaces in a string in Python. For this, you can invoke the **count()** method on the original string and pass the space character as input to the **count()** method as shown in the following example.

```
>>> myStr = "Count, the number of spaces"
>>> print("The string is:",myStr)
>>> character=" "
>>> position=myStr.count(character)
>>> print("The number of spaces in the string is:",position)
```

Output:

The string is: Count, the number of spaces

The number of spaces in the string is: 4

String Slicing in Python

To perform string manipulation in Python, you can use the syntax *string_name*[*start_index : end_index*] to get a substring of a string. Here, the slicing operation gives us a substring containing characters from *start_index* to *end_index-1* of the string *string_name*.

Keep in mind that python, as many other languages, starts to count from 0!!

```
>>> word = "Hello World"
>>> print word[0] #get one char of the word
>>> print word[0:1] #get one char of the word (same as above)
```

```
>>> print word[0:3] #get the first three char
>>> print word[:3] #get the first three char
>>> print word[-3:] #get the last three char
>>> print word[3:] #get all but the three first char
>>> print word[:-3] #get all but the three last character
```

```
>>> word = "Hello World"
>>> word[start:end] # items start through end-1
>>> word[start:] # items start through the rest of the list
>>> word[:end] # items from the beginning through end-1
>>> word[:] # a copy of the whole list
```

To learn how all the statements in the above code work, read this article on string slicing in Python. You can also have a look at this article on string splicing.

Split a String in Python

You can split a string using the split() method to perform string manipulation in Python. The **split()** method, when invoked on a string, takes a character as its input argument. After execution, it splits the string at the specified character and returns a list of substrings as shown below.

```
>>> word = "Hello World"
>>> print(word.split(' ')) # Split on whitespace
```

Output:

```
['Hello', 'World']
```

In the above example, we have split the string at the space character. Hence, the **split()** method returns a list containing two words i.e. 'Hello' and 'World' that are separated by the space character in the original string. You can pass any character to the **split()** method to split the string.

If the character passed to the **split()** method isn't present in the original string, the **split()** method returns a list containing the original string. You can observe this in the following example.

```
>>> word = "Hello World"
>>> print(word.split('Z')) # Split on whitespace
```

Output:

```
['Hello World']
```

In the above example, we passed the character "Z" to the **split()** method. As the character "Z" is absent in the string 'Hello World', we get a list containing the original string.

Check if a String Starts With or Ends With a Character in Python

To check if a string starts with or ends with a specific character in Python, you can use the startswith() or the endswith() method respectively.

- The startswith() method, when invoked on a string, takes a character as input argument. If the string starts with the given character, it returns True. Otherwise, it returns False.
- The endswith() method, when invoked on a string, takes a character as input argument. If the string ends with the given character, it returns True. Otherwise, it returns False. You can observe this in the following example.

```
aditya1117@aditya1117-Inspiron-15-3567:~$ python3
Python 3.10.6 (main, May 29 2023, 11:10:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> word = "hello world"
>>> word.startswith("H")
False
>>> word.endswith("d")
True
>>> word.endswith("w")
False
>>> word.startswith("h")
True
>>>
```

String Manipulation in Python

In the above example, you can observe that the string "hello world" starts with the character "h". Hence, when we pass the character "h" to the startswith() method, it returns True. When we pass the character "H" to the startswith() method, it returns False.

In a similar manner, the string "hello world" ends with the character "d". Hence, when we pass the character "d" to the endswith() method, it returns True. When we pass the character "w" to the endswith() method, it returns False.

Repeat Strings Multiple Times in a String in Python

You can repeat a string multiple times using the multiplication operator. When we multiply any given string or character by a positive number N, it is repeated N times. You can observe this in the following example.

```
>>> myStr="PFB "
>>> myStr*5
'PFB PFB PFB PFB PFB '
>>>
```

Repeat String

In the above example, you can observe that the string "PFB " is repeated 5 times when we multiply it by 5.

Replace Substring in a String in Python

You can replace a substring with another substring using the replace() method in Python. The replace() method, when invoked on a string, takes the substring to replaced as its first input argument and the replacement string as its second input argument. After execution, it replaces the specified substring with the replacement string and returns a modified string.

You can perform string manipulation in Python using the replace() method as shown below.

```
>>> word
'Hello World'
>>> word.replace("Hello", "Goodbye")
'Goodbye World'
>>> word
'Hello World'
>>>
```

Replace Substring in a string

In the above example, you can observe that we have replaced the substring "Hello" from the original string. The replace() method returns a new string with the modified characters.

In the output, you can observe that the original string remains unaffected after executing the replace() method. Hence, you can replace a substring in a string in Python and create a new string. However, the original string remains unchanged.

Changing Upper and Lower Case Strings in Python

You can convert string into uppercase, lowercase, and title case using the upper(), lower(), and title() method.

- The upper() method, when invoked on a string, changes the string into uppercase and returns the modified string.
- The lower() method, when invoked on a string, changes the string into lowercase and returns the modified string.
- The title() method, when invoked on a string, changes the string into titlecase and returns the modified string.
- You can also capitalize a string or swap the capitalization of the characters in the string using the capitalize() and the swapcase() method.
    - The capitalize() method, when invoked on a string, capitalizes the first character of the string and returns the modified string.
    - The swapcase() method, when invoked on a string, changes the lowercase characters into uppercase and vice versa. After execution, it returns the modified string.

You can observe these use cases in the following example.

```
>>> string = "Hello World"
>>> string.upper()
'HELLO WORLD'
>>> string.lower()
'hello world'
>>> string.title()
'Hello World'
>>> string.capitalize()
'Hello world'
>>> string.swapcase()
'hELLO wORLD'
>>>
```

String Case Changes

Reverse a String in Python

To reverse a string in Python, you can use the reversed() function and the join() method.

- The reversed() function takes a string as its input argument and returns a list containing the characters of the input string in a reverse order.
- The join() method, when invoked on a separator string, takes a list of characters as its input argument and joins the characters of the list using the separator. After execution, it returns the resultant string.

To reverse a string using the reversed() function and the join() method, we will first create a list of characters in reverse order using the reversed() function. Then we will use an empty string as

a separator and invoke the join() method on the empty string with the list of characters as its input argument.

After execution of the join() method, we will get a new reversed string as shown below.

```
string = "Hello World"

print(''.join(reversed(string)))
```

Output:

```
dlroW olleH
```

In the above example, you can observe that the original string is printed in the reverse order. Instead of the reversed() function and the join() method, you can also use the indexing operator to reverse a string as shown below.

```
string = "Hello World"

print(string[::-1])
```

Output:

```
'dlroW olleH'
```

Strip a String in Python

Python strings have the strip(), lstrip(), rstrip() methods for removing any character from both ends of a string.

- The strip() method when invoked on a string, takes a character as its input argument and removes the character from start (left) and end(right) of the string. If the characters to be removed are not specified then white-space characters will be removed.
- The lstrip() method when invoked on a string, takes a character as its input argument and removes the character from start (left) of the string.
- The rstrip() method when invoked on a string, takes a character as its input argument and removes the character from the end(right) of the string.

We can strip the * characters from a given string using the methods as shown below.

```
>>> myStr="*****PFB*****"
>>> myStr
'*****PFB*****'
>>> myStr.lstrip("*")
'PFB*****'
>>> myStr.rstrip("*")
'*****PFB'
>>> myStr.strip("*")
'PFB'
>>>
```

Strip String in Python

In the above example, the original string contains "*" character at its start and end. Hence, we need to pass the "*" character to the strip(), lstrip() or rstrip() method to strip it from the given string.

When we do not pass a character to the strip(), lstrip() or rstrip() method, they take the space character as its value and strips the string on which they are invoked. You can observe this in the following example.

```
>>> myStr="    PFB    "
>>> myStr
'    PFB    '
>>> myStr.lstrip()
'PFB    '
>>> myStr.rstrip()
'    PFB'
>>> myStr.strip()
'PFB'
>>>
```

Strip String in Python

Concatenate Strings in Python

To concatenate strings in Python, you can use the “+” operator as shown below.

```
>>> str1="Hello "
>>> str2="World"
>>> str3=str1+str2
>>> str3
'Hello World'
>>>
```

Concatenate Strings

You can concatenate two or more strings using the + operator as shown above.

Check Properties of a String in Python

A string in Python can be tested for Truth value. For this, we can use different methods to check the properties of the string. The return type of these methods is of Boolean type (True or False).

Here is a list of some of the methods to check properties of a string in Python.

word = "Hello World"

word.isalnum() #check if all char are alphanumeric

word.isalpha() #check if all char in the string are alphabetic

word.isdigit() #test if string contains digits

word.istitle() #test if string contains title words

word.isupper() #test if string contains upper case

word.islower() #test if string contains lower case

word.isspace() #test if string contains spaces

word.endswith('d') #test if string endswith a d

word.startswith('H') #test if string startswith H

Conclusion

In this article, we have discussed different ways to perform string manipulation in Python. To learn more about python programming, you can read this article on list comprehension in Python. You might also like this article on how to build a chatbot in python.

I hope you enjoyed reading this article. Stay tuned for more informative articles.

Happy Learning!

Related

Python String Methods for String ManipulationMay 6, 2021In "Basics"

Python – Quick GuideMarch 9, 2016In "Basics"

Convert String to DataFrame in PythonMarch 20, 2023In "Basics"

Recommended Python Training

Course: Python 3 For Beginners

Over 15 hours of video content with guided instruction for beginners. Learn how to create real world applications and master the basics.

Enroll Now

Filed Under: Basics, Python Strings Author: PFB Staff Writer

More Python Topics

API Argv Basics Beautiful Soup Cheatsheet Code Code Snippets Command Line Comments Concatenation crawler Data Structures Data Types deque Development Dictionary Dictionary Data Structure In Python Error Handling Exceptions Filehandling Files Functions Games GUI Json Lists Loops Mechanzie Modules Modules In Python Mysql OS pip Pyspark Python Python On The Web Python Strings Queue Requests Scraping Scripts Split Strings System & OS urllib2

Primary Sidebar

Search this website

Menu

- Basics
- Cheatsheet

- Code Snippets
- Development
- Dictionary
- Error Handling
- Lists
- Loops
- Modules
- Scripts
- Strings
- System & OS
- Web

Get Our Free Guide To Learning Python

Email Address

Most Popular Content

- Reading and Writing Files in Python
- Python Dictionary – How To Create Dictionaries In Python
- How to use Split in Python
- Python String Concatenation and Formatting
- List Comprehension in Python
- How to Use sys.argv in Python?
- How to use comments in Python
- Try and Except in Python

Recent Posts

- Count Rows With Null Values in PySpark
- PySpark OrderBy One or Multiple Columns
- Select Rows with Null values in PySpark
- PySpark Count Distinct Values in One or Multiple Columns
- PySpark Filter Rows in a DataFrame by Condition

Part II: Math for Areas and Volumes of Shapes

1. Introduction

In this exercise you will write several Python program files that calculate the area and volume of different geometric shapes.

2. Objectives

The purpose of this assignment is to give you experience with:

- Using program files and mathematical formulas in Python.

- Converting mathematical expressions into Python expressions.

- Doing input and output at the command line.

- Getting some exposure to strings.

- Doing a bit of independent research. (That may not be as much fun as it sounds — in this case it just means figuring stuff out on your own.)

3. Background

3.1. The Python prompt

In this document I have you type expressions into Python. Start Linux and log in. When I show you an expression that starts with a prompt like this:

>>>

Then that means you should type the expression into Python. For example, this expression:

>>> 1 + 2

means that you should type the part that I have underlined, meaning the expression 1 + 2, into Python, but not the >>> part.

3.2. Float and int

There are two types of numbers in Python, **ints** and **floats**. An **int** (integer) is a whole number with no fractional part. A **float** (floating point number) is a number that may have a fractional part. Python makes a distinction between the two types because the computer itself makes a distinction between the two types internally. Doing arithmetic only with **int**s is done differently, and a bit more efficiently, than arithmetic with **float**s.

There is a function called **type** in Python that will tell you what kind of number you're looking at. Give it a try:

```
>>> type(1)
<class 'int'>
```

That means that the number 1 is treated as an **int**.

```
>>> type(1.5)
<class 'float'>
```

You can convert easily between the numeric types:

```
>>> float(1)
1.0

>>> int(1.5)
1
```

Notice that converting a float to an int may cause a loss of numeric accuracy: the fractional part is simply truncated. There is a **round** function that will round to the nearest whole number:

```
>>> round(1.5)
2
```

You can also covert a string to a number using either the **int** or **float** function.

```
>>> int('100')
100

>>> float('100')
100.0
```

But this doesn't work:

```
>>> int('100.5')
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '100.5'
```

It's ok to convert a string that looks like an int into an int, but it won't convert a string that looks like a **float** into an **int**.

3.3. The input function

The input function in Python is a really nice way to let the user enter information into your program. Start Python and try it out a few times.

```
>>> input()
abc
'abc'
```

What happened there was that I entered the **input()** function call, Python stopped and waited for me to enter a line. I typed **abc** and then hit Enter. It showed me what the result of that was: the string **'abc'**.

Notice that the string was shown with quotation marks around it. That's because Python showed me that string as the result of an expression, not as the output from a **print** statement.

It's more useful to use an assignment statement to save the result of the **input()** function call so that you can use it later.

```
>>> x = input()
abc
>>> x
'abc'
```

You can also enter a number:

```
>>> x = input()
123
>>>
```

Notice that nothing was displayed after you entered 123. That's because Python does not display anything as the result of an assignment statement. You can ask Python to show you the value that it stored in the variable x:

```
>>> x
'123'
```

Hmm, the number has quotes around it which means that it's a string and not actually a number. We'll deal with that shortly.

You can also supply a string as an argument to the input function and it will display the string as a prompt to the user:

```
>>> x = input('Please enter a number: ')
Please enter a number: 123
```

Like I said earlier, although the number looks like a number, it's actually a string. Let's try

to use that number in a math expression:

```
>>> x + 1
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

It's important to see error messages like this so that you can start to learn what they mean. Here it's trying to convert an int into a str, which means an integer (number) into a string. You can use the + operator to join two strings together, but Python does not let you do math using a string and a number, even if the string looks like a number.

3.4. Converting input to a float

In this lab exercise you will be doing math using floats, so the number that the input function returns should be converted from a string to a float. There are a few ways you could do that.

1. You can call the **float** function directly on the result of the **input** function:

```
>>> x = float(input('Please enter a number: ') )
                  \        \_____________________/ /
                   \_____________________________ /
```

Be mindful of the parentheses: The pairs must match up.

2. You can call the **float** function on the variable after the input function:

```
>>> x = input('Please enter a number: ')
>>> x = float(x)
```

Yes, I'm using **x** twice in that second line: once on the right hand side as an argument to the **float** function, and once on the left hand side as the destination variable of the assignment statement. Keep typing:

```
>>> x
123.0
>>> x + 1
124.0
```

3. If that's confusing then you can introduce a second variable:

```
>>> x = input('Please enter a number: ')
>>> y = float(x)
```

But now when you're doing math you must use y instead of x, because x still contains a string.

3.5. Using math.pi

In this exercise you'll be calculating the area of a circle, and to do that you'll need to use the value of π. You could simply type the number 3.14, or 3.14159, or 3.14159265358979, but Python already has a variable that contains the value of π., and it's called math.pi. Try this:

```
>>> math.pi
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
NameError: name 'math' is not defined
```

Ooops. If any name in Python is segmented with a dot like that, then the first part of the name must be imported before the name can be used:

```
>>> import math
>>> math.pi
3.141592653589793
```

Now it works.

4. Assignment

Create a new folder called Lab01. All the files you create for this lab exercise should be created inside that folder.

4.1. Area of a circle

Below I give you a template for a Python program file that calculates the area of a circle.

Enter this program into a file called circle.py by first starting your editor of choice along with the name of the file to edit:

Example using gedit. If you are using a specific IDE, then your methods for creating and saving Python scripts will vary.

$ gedit circle.py

Type the following statements into gedit:

```
import math
radius = input('Enter radius: ')
area = 0
print('The area of a circle with radius', radius, 'is', area)
```

Yes, copy & paste should work fine as long as you've completed the exercise that I just sent you on using remote logins, but please type these statements, anyway. You need to get used to typing Python.

Any import statements that your program needs should be the first thing in the program. The input function on the second line causes the program to pause in order to let the user enter a number. Because the input function is used on the RHS (right-hand side) of an assignment statement, the number that the user enters will be placed into the variable called radius.

Your job is to replace the 0 on the right side of the area = 0 assignment statement with the formula for the area a of a circle having radius r, which is this:

$$a = \pi r^2$$

The program above uses variables named area and radius instead of just a and r.

There are a few more things you need to know to write this formula in Python.

- You will need to convert the radius into a float. Use one of the ways I showed you in the previous section. If you introduce a new variable, then be sure to use that new variable in your math expression.
- You must use the * character for multiplication. The product of two expressions ab in Python must be written a * b.
- You must use the ** operator to raise a number to a power. The power x2 in Python must be written x ** y.

I know that many of you don't know how to write programs yet and you might be pretty confused by all this, but you'll get it soon. I promise.

If you're not sure what to do, give it your best guess. Ask for help from your TA or from a neighbor if you can't figure it out, or see if you can get some answers from Google.

Test it

Whenever you finish writing a program file you should test it. To run this program file, exit nano and then type python3 circle.py at the command line. Python will run the statements found in that file. Make sure your output shows the correct answers, shown below.

```
$ python circle.py
Enter radius: 10
('The area of a circle with radius', 10, 'is', 314.1592653589793)
```

Hmm, that doesn't look right! I typed python instead of python3, so it ran Python 2 and showed something other than what it should be. The information is correct but it's not formatted correctly. Try it again with Python 3:

```
$ python3 circle.py
Enter radius: 10
The area of a circle with radius 10 is 314.1592653589793
```

That worked! Try a few more. Use the up-arrow to retrieve the last command that you entered at the command prompt so that you don't have to re-type it.

```
$ python3 circle.py
Enter radius: 12.345

The area of a circle with radius 12.345 is 478.7756573542473
$ python3 circle.py
Enter radius: 0.1
The area of a circle with radius 0.1 is 0.031415926535897934
```

4.2. Area of a square

Create a square.py file and write a program that calculates the area of a square based on the user's input. You can copy the circle.py file to start with, then change it so that it works like I show you below. This command copies the circle.py file into a new file called square.py:

```
$ cp circle.py square.py
```

Now edit the square.py file:

```
$ nano square.py
```

Change the formula. The area of a square is the length of one of the sides squared.

$$a = s^2$$

You will need to re-name the radius variable to something more meaningful. I suggest **side**.

Also, you can delete the **import math** statement from the top of the file, since you don't need to use **pi** anymore.

Don't forget to convert **side** into a **float** somehow.

Here's how my program runs:

```
$ python3 square.py
Enter side: 3
Traceback (most recent call last):
  File "square.py", line 5, in <module>
    print('The area of a square with side', radius, 'is', area)
NameError: name 'radius' is not defined
```

Oops. I forgot to change the **radius** variable inside the **print** statement to **side**.

```
$ python3 square.py
Enter side: 3
The area of a square with side 3 is 9.0
```

```
$ python3 square.py
Enter side: 3.14159
The area of a square with side 3.14159 is 9.869587728099999
```

4.3. Area of a rectangle

Write a program called **rectangle.py** and have it work similar to **circle** and **square**.

This is the formula for the area of a rectangle:

$$a = lw$$

where l = length of rectangle,
w = width of rectangle

Here you will need to use two **input** statements, like this:

```
length = input( )
width = input( )
```

Here's the output from my program:

```
Enter length of the rectangle: 3
```

Enter the width of the rectangle: 5
The area of a rectangle with length 3 and width 5 is 15.0

That last print statement will be more complex. You should type it like this: (I've abbreviated so it will fit on the page)

print('... with length', length, 'and width', width, 'is', area)
___/ __/ __/

See how I'm using the variables?

It's ok if your program displays **length 3.0** and **width 5.0**. instead of **length 3** and **width 5**.

4.4. Area of a triangle

Calculating the area of a triangle requires knowing the width of the base (known just as the base) and the height. The area of the triangle is calculated using this expression:

$$a = ½\ bh$$

Write a program file called **triangle.py** that works similar to the other programs and calculates the area of a triangle. You may copy the **rectangle.py** file if you think that will be easier.

To multiply something like x by a fraction in Python, you may use an expression like this:

```
area = 1/2 * x
```

That expression looks like "one half times x" but it's actually just "one divided by two times x", which is actually the same thing. You can also do this:

```
area = x / 2
```

which is algebraically equivalent.

Run it.

Enter base of the triangle: 5
Enter the height of the triangle: 9

The area of a triangle with base 5 and height 9 is 22.5

It's ok if your program displays **base 5.0** and **height 9.0**. instead of **base 5** and **height 9**.

4.5. Volume of a sphere

Write a program file that calculates the volume of a sphere. Name it **sphere.py**. This program file will be similar to the **circle.py** program file.

Use the interwebs to search for the formula of a sphere and use it in this program file. Run the program to get some output, and then verify it using some other means in order to determine that your program generates the correct output values. You can even use Google to do the math for you, but you must use strict Python-style math expressions. Copy and paste this into the Google search box to see what I mean:

4 / 3 * pi * 3 ^ 3

4.6. Volume of a cone

Call it **cone.py**. You know what to do, right? You may assume that the cone is a *right circular cone*.

4.7. Volume of a box

Calculate the volume of a rectangular box having a height, width, and depth.