CSC462 Artificial Intelligence

LAB 3: The Python REPL

Introduction

In this lab, you will learn how to write and run your first lines of Python code at the Python REPL also called the Python prompt. You will learn how to use Python as a calculator and be introduced to Python variables and Python's `print()` function.

By the end of this lab, you will be able to:

- Open and close the Python REPL
- Compute mathematical calculations using the Python REPL
- Use the output from the Python REPL as input in another problem
- Import the math and statistics modules from the Python Standard Library and use their functions
- Assign values to variables
- Use variables in calculations
- Create strings
- Combine and compare strings

Python as a Calculator

Python can be used as a calculator to compute arithmetic operations like addition, subtraction, multiplication and division. Python can also be used for trigonometric calculations and statistical calculations.

Arithmetic

Python can be used as a calculator to make simple arithmetic calculations.
Simple arithmetic calculations can be completed at the Python Prompt, also called the *Python REPL*. REPL stands for Read Evaluate Print Loop. The Python REPL shows three arrow symbols `>>>` followed by a blinking cursor. Programmers type commands at the `>>>` prompt then hit `[ENTER]` to see the results.
Commands typed into the Python REPL are *read* by the interpreter, results of running the commands are *evaluated*, then *printed* to the command window. After the output is printed, the `>>>` prompt appears on a new line. This process repeats over and over again in a continuous *loop*.

Try the following commands at the Python REPL:

Suppose the mass of a battery is 5 kg and the mass of the battery cables is 3 kg. What is the mass of the battery cable assembly?

```
>>> 5 + 3
8
```

Suppose one of the cables above is removed and it has a mass of 1.5 kg. What is the mass of the leftover assembly?

```
>>> 8 - 1.5
```

```
    6.5
```

If the battery has a mass of 5000 g and a volume of 2500 ml What is the density of the battery? The formula for density is below, where d is density, m is mass and v is volume.

In the problem above and

Let's solve this with Python.

```
    >>> 5000 / 2500
    2.0
```

What is the total mass if we have 2 batteries, and each battery weighs 5 kg?

```
    >>> 5 * 2
    10
```

The length, width, and height of each battery is 3 cm. What is the area of the base of the battery? To complete this problem, use the double asterisk symbol `**` to raise a number to a power.

```
    >>> 3 ** 2
    9
```

What is the volume of the battery if each the length, width, and height of the battery are all 3 cm?

```
    >>> 3 ** 3
    27
```

Find the mass of the two batteries and two cables.
We can use Python to find the mass of the batteries and then use the answer, which Python saves as an underscore _ to use in our next operation. (The underscore _ in Python is comparable to the `ans` variable in MATLAB)

```
    >>> 2 * 5
    10
    >>> _ + 1.5 + 1
    12.5
```

Section Summary

A summary of the arithmetic operations in Python is below:

Operator	**Description**	**Example**	**Result**
+	addition	`2 + 3`	`5`
-	subtraction	8 - 6	2
-	negative number	-4	-4
*	multiplication	5 * 2	10
/	division	6 / 3	2

Operator	Description	Example	Result
**	raises a number to a power	10**2	100
_	returns last saved value	_ + 7	107

Trigonometry: sine, cosine, and tangent

Trigonometry functions such as sine, cosine, and tangent can also be calculated using the Python REPL.

To use Python's trig functions, we need to introduce a new concept: *importing modules*.

In Python, there are many operations built into the language when the REPL starts. These include `+` , `-`, `*`, `/` like we saw in the previous section. However, not all functions will work right away when Python starts. Say we want to find the sine of an angle. Try the following:

```
>>> sin(60)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sin' is not defined
```

This error results because we have not told Python to include the `sin` function. The `sin` function is part of the *Python Standard Library*. The Python Standard Library comes with every Python installation and includes many functions, but not all of these functions are available to us when we start a new Python REPL session. To use Python's `sin` function, first import the `sin` function from the `math` *module* which is part of the Python Standard Library.

Importing modules and functions is easy. Use the following syntax:

```
from module import function
```

To import the `sin()` function from the `math` module try:

```
>>> from math import sin
>>> sin(60)
-0.3048106211022167
```

Success! Multiple modules can be imported at the same time. Say we want to use a bunch of different trig functions to solve the following problem.

An angle has a value of /6 radians. What is the sine, cos, and tangent of the angle?

To solve this problem we need to import the `sin()`, `cos()`, and `tan()` functions. It is also useful to have the value of , rather than having to write `3.14...`. We can import all of these functions at the same time using the syntax:

```
from module import function1, function2, function3
```

Note the commas in between the function names.

Try:

```
>>> from math import sin, cos, tan, pi
>>> pi
3.141592653589793
>>> sin(pi/6)
0.49999999999999994
>>> cos(pi/6)
0.8660254037844387
>>> tan(pi/6)
```

```
0.5773502691896257
```

Section Summary

The following trig functions are part of Python's **math** module:

Trig function	**Name**	**Description**	**Example**	**Result**
math.pi	pi	mathematical constant	math.pi	3.14
math.sin()	sine	sine of an angle in radians	math.sin(4)	9.025
math.cos()	cosine	cosine of an angle in radians	cos(3.1)	400
math.tan()	tangent	tangent of an angle in radians	tan(100)	2.0
math.asin()	arc sine	inverse sine, ouput in radians	math.sin(4)	9.025
math.acos()	arc cosine	inverse cosine, ouput in radians	log(3.1)	400
math.atan()	arc tangent	inverse tangent, ouput in radians	atan(100)	2.0
math.radians()	radians conversion	degrees to radians	math.radians(90)	1.57
math.degrees()	degree conversion	radians to degrees	math.degrees(2)	114.59

Exponents and Logarithms

Calculating exponents and logarithms with Python is easy. Note the exponent and logarithm functions are imported from the **math** module just like the trig functions were imported from the **math** module above.

The following exponents and logarithms functions can be imported from Python's math module:

- `log`
- `log10`
- `exp`
- `e`
- `pow(x,y)`
- `sqrt`

Let's try a couple of examples:

```
>>> from math import log, log10, exp, e, pow, sqrt
>>> log(3.0*e**3.4)          # note: natural log
4.4986122886681095
```

A right triangle has side lengths 3 and 4. What is the length of the hypotenuse?

```
    >>> sqrt(3**2 + 4**2)
    5.0
```

The power function `pow()` works like the `**` operator. `pow()` raises a number to a power.

```
    >>> 5**2
    25

    >>> pow(5,2)
    25.0
```

Section Summary

The following exponent and logarithm functions are part of Python's **math** module:

Math function	Name	Description	Example	Result
math.e	Euler's number	mathematical constant	math.e	2.718
math.exp()	exponent	raised to a power	math.exp(2.2)	9.025
math.log()	natural logarithm	log base e	math.log(3.1)	400
math.log10()	base 10 logarithm	log base 10	math.log10(100)	2.0
math.pow()	power	raises a number to a power	math.pow(2,3)	8.0
math.sqrt()	square root	square root of a number	math.sqrt(16)	4.0

Statistics

To round out this section, we will look at a couple of statistics functions. These functions are part of the Python Standard Library, but not part of the **math** module. To access Python's statistics functions, we need to import them from the **statistics** module using the statement `from statistics import mean, median, mode, stdev`. Then the functions `mean`, `median`, `mode` and `stdev` (standard deviation) can be used.

```
    >>> from statistics import mean, median, mode, stdev

    >>> test_scores = [60, 83, 83, 91, 100]

    >>> mean(test_scores)
    83.4

    >>> median(test_scores)
    83

    >>> mode(test_scores)
    83

    >>> stdev(test_scores)
    14.842506526863986
```

Alternatively, we can import the entire **statistics** module using the statement `import statistics`. Then to use the functions, we need to use the names `statistics.mean`, `statistics.median`, `statistics.mode`, and `statistics.stdev`. See below:

```
>>> import statistics

>>> test_scores = [60, 83, 83, 91, 100 ]

>>> statistics.mean(test_scores)
83.4

>>> statistics.median(test_scores)
83

>>> statistics.mode(test_scores)
83

>>> statistics.stdev(test_scores)
14.842506526863986
```

Section Summary

The following functions are part of Python's **statistics** module. These functions need to be imported from the `statistics` module before they are used.

Statistics function	**Name**	**Description**	**Example**	**Result**
mean()	mean	mean or average	mean([1,4,5,5])	3.75
median()	median	middle value	median([1,4,5,5])	4.5
mode()	mode	most often	mode([1,4,5,5])	5
stdev()	standard deviation	spread of data	stdev([1,4,5,5])	1.892
variance()	variance	variance of data	variance([1,4,5,5])	3.583

Variables

Variables are assigned in Python using the `=` equals sign also called the *assignment operator*. The statement:

```
a = 2
```

Assigns the integer `2` to the variable `a`.

```
>>> a = 2
>>> a
2
```

Note the assignment operator =(equals), is different from the logical comparison operator == (equivalent to).

```
>>> a == 2
```

```
True
```

Variable names in Python must conform to the following rules:

- variable names must start with a letter
- variable names can only contain letters, numbers, and the underscore character _
- variable names can not contain spaces
- variable names can not include punctuation
- variable names are not enclosed in quotes or brackets

The following code lines show valid variable names:

```
constant = 4

new_variable = 'var'

my2rules = ['rule1','rule2']

SQUARES = 4
```

The following code lines show invalid variable names:

```
a constant = 4

3newVariables = [1, 2, 3]

&sum = 4 + 4
```

Let's solve the problem below at the Python REPL using variables.

Problem

The Arrhenius relationship states:

In a system where , , , and , calculate .

Solution

Use variables to assign a value to each one of the constants in the problem and calculate .

```
>>> nv = 2.0e-3
>>> Qv = 5
>>> R = 3.18
>>> T = 293
>>> from math import exp
>>> n = nv*exp(-1*Qv/(R*T))
>>> n
0.0019892961379660424
```

String Operations

Strings are sequences of letters, numbers, punctuation, and spaces. Strings are defined at the Python REPL by enclosing letters, numbers, punctuation, and spaces in single quotes `' '` or double quotes `" "`.

```
>>> word = "Solution"
>>> another_word = "another solution"
>>> third_word = "3rd solution!"
```

In Python, string operations include concatenation (combining strings), logical comparisons (comparing strings) and indexing (pulling specific characters out of strings).

String Concatenation

Strings can be *concatenated* or combined using the + operator.

```
>>> word = "Solution"
>>> another_word = "another solution"
>>> third_word = "3rd solution!"
>>> all_words = word+another_word+third_word
>>> all_words
'Solutionanother solution3rd solution!'
```

To include spaces in the concatenated string, add a string which just contains one space " " in between each string you combine.

```
>>> word = "Solution"
>>> another_word = "another solution"
>>> third_word = "3rd solution!"
>>> all_words = word + " " + another_word + " " + third_word
>>> all_words
'Solution another solution 3rd solution!'
```

String Comparison

Strings can be compared using the comparison operator; the double equals sign ==. Note the comparison operator (double equals ==) is not the same as the assignment operator, a single equals sign =.

```
>>> name1 = 'Gabby'
>>> name2 = 'Gabby'
>>> name1 == name2
True
>>> name1 = 'Gabby'
>>> name2 = 'Maelle'
>>> name1 == name2
False
```

Capital letters and lower case letters are different characters in Python. A string with the same letters, but different capitalization are not equivalent.

```
>>> name1 = 'Maelle'
>>> name2 = 'maelle'
>>> name1 == name2
False
```

Print Statements

One built-in function in Python is `print()`. The value or expression inside of the parenthesis of a `print()` function "prints" out to the REPL when the `print()` function is called.

An example using the `print()` function is below:

```
>>> name = "Gabby"
>>> print("Your name is: ")
Your name is:
>>> print(name)
Gabby
```

Remember that strings must be enclosed by quotation marks. The following command produces an error.

```
>>> print(Gabby)

NameError: name 'Gabby' is not defined
```

This error is corrected by surrounding the string `Gabby` with quotation marks.

```
>>> print("Gabby")
Gabby
```

Expressions passed to the `print()` function are evaluated before they are printed out. For instance, the sum of two numbers can be shown with the `print()` function.

```
>>> print(1+2)
3
```

If you want to see the text `1+2`, you need to define `"1+2"` as a string and print out the string `"1+2"` instead.

```
>>> print("1+2")
1+2
```

Strings can be concatenated (combined) inside of a `print()` statement.

```
>>> name = Gabby
>>> print('Your name is: ' + name)
Your name is Gabby
```

The `print()` function also prints out individual expressions one after another with a space in between when the expressions are placed inside the `print()` function and separated by a comma.

```
>>> print("Name:","Gabby","Age", 2+7)
Name: Gabby Age 9
```

Summary

In this lab, you learned how to use the Python REPL, also called the Python prompt, to solve problems. You learned how to do arithmetic, powers and logarithms, trigonometry and save values to variables. Operations on strings were introduced including concatenation and comparison. In the last section of the chapter, Python's `print()` function was introduced.

Summary of Python Functions and Commands

Below is a summary of the functions and operators used in this chapter:

Arithmetic

Arithmetic Operator	**Description**
`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponents

Arithmetic Operator	Description
_	answer in memory

Trigonometry

Trig Function	Description
`from math import *`	
`sin`	sine of angle in radians
`cos`	cosine of angle in radians
`tan`	tangent of angle in radians
`pi`	
`degrees`	convert radians to degrees
`radians`	convert degrees to radians
`asin`	inverse sine
`acos`	inverse cosine
`atan`	inverse tangent

Logarithms and Exponents

Logarithms and Exponent Function	Description
`from math import *`	
`log`	log base e, natural log

Logarithms and Exponent Function	**Description**
`log10`	log base 10
`exp`	
`e`	the math constant
`pow(x,y)`	x raised to the y power
`sqrt`	square root

Statistics

Statistics Function	**Description**
`from statistics import *`	
`mean`	mean (average)
`median`	median (middle value)
`mode`	mode (most often)
`stdev`	standard deviation of a sample
`pstdev`	standard deviation of a population

```
In [ ]:
```