Lecture 5
Math module

Objectives of this Lecture

- A little revision
- To learn how to use the Python math library
- Example: Quadratic equation
- To understand the accumulator program
- Example: Factorial

REVISION: The Software Development Process

1. Analyse the Problem
2. Determine the Specifications
3. Create a Design
4. Implement the Design i.e. write the program
5. Test/Debug the Program
6. Maintain the Program

Software Life Cycle

REVISION: Numerical data types

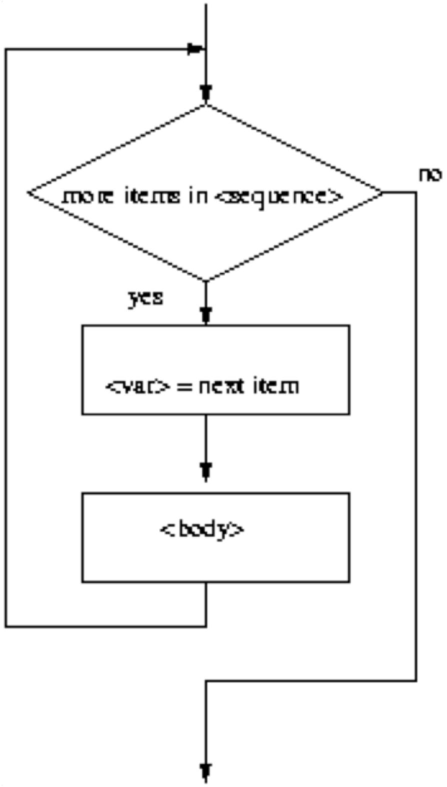
Conversion function	Example use	Value returned
int(<a number or string>)	int(2.87)	2
int	int("55")	55
float(<a number or string>)	float(32)	32.0
str(<any value>)	str(43)	'43'

What happens if you type: `int("2.87")`?

REVISION: Definite Loops

`for` loops alter the flow of program execution, so they are referred to as control structures.

```
>>> for odd in [1, 3, 5, 7]:
        print(odd*odd)
1
9
25
49
```



Loop variable odd first has the value 1, then 3, then 5 and finally 7

REVISION: module and function

- Module:
  - allows you to logically organize your Python code
  - grouping related code that makes the code easier to understand and use
  - Simply, a file consisting of Python code which can define functions, classes, variables and may also include runnable code
- Function:
  - a block of organized, reusable code that is used to perform a single, related action.
  - provide better modularity for your application and a high degree of code reusing

Using the Math Library

- Besides (**, *, /, %, //, +, -, abs), we have lots of other math functions available in a *math library*.
  - `**` *is exponentiation, e.g.* `2**3 == 8`

- A *library* is a module with some useful definitions/functions.

Using the Math Library

- The formula for computing the roots of a quadratic equation of the form: $ax^2 + bx + c$ is

$$root = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- The only part of this that we don't know how to do yet is the square root function, but that is in the math library

Using the Math Library

- To use a library, we need to make sure this line is in our program:

```
import math     # not maths
```

- Importing a library makes whatever functions are defined within it available to the program (from that point onwards).

Using the Math Library

- To access the sqrt library routine, we need to access it as: `math.sqrt(x)`

- Using this dot notation tells Python to use the sqrt function found in the math library module.

- To calculate the root, you can do
```
discRoot = math.sqrt(b*b - 4*a*c)
```

Using the Math Library

- We can also import math module as

```
import math as m # now math is called by m
m.sqrt(9)
```

- If we do not want to import all functions of module

```
# Import sqrt function only
from math import sqrt
sqrt(9) # No need a reference
```

Using the Math Library

```
# quadratic.py
#    A program that computes the real roots of a quadratic equation.
#    Illustrates use of the math library.
#    Author: Unit Coordinator


import math  # Makes the math library available.


def main():
    print("This program finds the real solutions to a quadratic  equation")
    print()

    a = float(input("Please enter coefficient a: "))
    b = float(input("Please enter coefficient b: "))
    c = float(input("Please enter coefficient c: "))

    discRoot = math.sqrt(b * b - 4 * a * c)
    root1 = (-b + discRoot) / (2 * a)
    root2 = (-b - discRoot) / (2 * a)

    print()
    print("The solutions are:", root1, root2 )
    return
```

Using the Math Library

Running the program

```
>>> main()
This program finds the real solutions to a quadratic
Please enter coefficient a: 3
Please enter coefficient b: 4
Please enter coefficient c: -1


The solutions are: 0.215250437022 -1.54858377035
```

Using the Math Library

Try again with 1, 2, 3

```
>>> main()
This program finds the real solutions to a quadratic
Please enter coefficient a: 1
Please enter coefficient b: 2
Please enter coefficient c: 3

Traceback (most recent call last):
  File "quadratic_roots.py", line 21, in <module> main()
  File "quadratic_roots.py", line 15, in main
    discRoot = math.sqrt(b * b - 4 * a * c)
ValueError: math domain error

>>>
```

Runtime error

Math Library

- If a = 1, b = 2, c = 3, then we are trying to take the square root of a negative number!
- Using the `sqrt` function is more efficient than using `**`
  - *How could you use* `**` *to calculate a square root?*

Math Library

Python	Mathematics	English
pi		An approximation of pi
e	e	An approximation of e
sqrt(x)		The square root of x
sin(x)	$\sin x$	The sine of x
cos(x)	$\cos x$	The cosine of x
tan(x)	$\tan x$	The tangent of x
asin(x)	$\arcsin x$	The inverse of sine x
acos(x)	$\arccos x$	The inverse of cosine x
atan(x)	$\arctan x$	The inverse of tangent x

Don’t forget to put `math` in front (depending on the method of importing module, e.g. `math.sqrt(x)`, `math.pi`

Math Library

Python	Mathematics	English
log(x)	$\ln x$	The natural (base e) logarithm of x
log10(x)	$\log_{10} x$	The common (base 10) logarithm of x
exp(x)	e^x	The exponential of x
ceil(x)	$\lceil x \rceil$	The smallest whole number >= x
floor(x)	$\lfloor x \rfloor$	The largest whole number <= x

Accumulating Results: Factorial

- Say you are waiting in a line with five other people. How many ways are there to arrange the six people?
- 720 -- which is the factorial of 6 (abbreviated 6!)
- Factorial is defined as:
  $$n! = n(n-1)(n-2)\ldots(1)$$
- So, 6! = 6*5*4*3*2*1 = 720

Accumulating Results: Factorial

- How we could write a program to do this?
- Input number to take factorial of, n
  Compute factorial of n, fact
  Output fact

Accumulating Results: Factorial

- How did we calculate 6!?
- 6*5 = 30
- Take that 30, and 30 * 4 = 120
- Take that 120, and 120 * 3 = 360
- Take that 360, and 360 * 2 = 720
- Take that 720, and 720 * 1 = 720

Algorithm

The general form of an accumulator algorithm looks like this:

- *Initialize the accumulator variable*
- *Perform computation*
  (e.g. in case of factorial multiply by the next smaller number)
- *Update accumulator variable*
- *Loop until final result is reached*
  (e.g. in case of factorial the next smaller number is 1)
- *Output accumulator variable*

Accumulating Results: Factorial

- It looks like we'll need a loop!

```
factorial = 1
for fact in [6, 5, 4, 3, 2, 1]:
    factorial = fact * factorial
```

- Let's trace through it to verify that this works!

Accumulating Results: Factorial

- Why did we need to initialize `factorial` to 1?
- There are a couple reasons…
  - *Each time through the loop, the previous value of* `factorial` *is used to calculate the next value of* `factorial`*. By doing the initialization, you know fact will have a value the first time through.*
  - *If you use* `factorial` *without assigning it a value, what does Python do?*

Improving Factorial

- What does `range(n)` return?
  0, 1, 2, 3, …, n-1

- `range` has another optional parameter:
  - `range(start, n)` *returns start, start + 1, …, n-1*
  - *E.g.* `range(1, 11)` *returns: 1,2,3,4,5,6,7,8,9,10*

- But wait! There's more!

  `range(start, n, step)`
  returns: `start, start+step,` …stopping before `n`

- list(<sequence>) to make a list

Range()

- Let's try some examples!

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5,10))
[5, 6, 7, 8, 9]
>>> list(range(5,10,2))
[5, 7, 9]
```

range() Forwards or Backwards

- We can do the range for our loop a couple of different ways.
  - *We can count up from 2 to n:*
    `range(2, n+1)`
    (Why did we have to use n+1?)
  - *We can count down from n to 2:*
    `range(n, 1, -1)`

Back at the Factorial Program

Our completed factorial program:

```
#    Program to compute the factorial of a number
#    Illustrates for loop with an accumulator
#    Author: Unit coordinator

def main():
    n = int(input("Please enter an integer: "))
    factorial = 1
    for fact in range(n,1,-1):
       factorial = fact * factorial
    print("The factorial of", n, "is", factorial)
    return

main()
```

Lecture Summary

- We learned how to use the Python `math` library
- We discussed an example of Quadratic equation
- We discussed an example of Factorial
- We discussed `range()` function