CSI31 Lecture 5

Topics:

3.1 Numeric Data Types
3.2 Using the Math Library
3.3 Accumulating Results: Factorial

3.1 Numeric Data Types

When computers were first developed, they were seen primarily as number crunchers.

data - is the information that is stored and manipulated by computer programs

Python built-in operations on numbers		
+	addition	a+b
-	subtraction	a-b
*	multiplication	a*b
/	division	a/b = a ÷ b
**	exponentiation	2**3 = 2^3 = 8
%	remainder	12 % 5 = 2
abs()	absolute value	abs(-23.4) = 23.4
//	integer division	12//5 = 2

3.1 Numeric Data Types

Different kinds of data will be stored and manipulated in different ways.

Integers: ..., -3, -2, -1, 0, 1, 2, 3, ... *integer data type (int)*
Decimals: 0.123, - 4.345, 7.1111 *floating point data type (float)*

try to input the following commands in Python shell:

```
>>> type(23)

>>>type(1.23)

>>>type(2.0)

>>>my_number=23.1
>>>type(my_number)
```

The function type *returns us the type of the value.*

3.1 Numeric Data Types

Different kinds of data will be stored and manipulated in different ways.

Integers: ..., -3, -2, -1, 0, 1, 2, 3, ...   *integer data type* (*int*)
Decimals: 0.123, - 4.345, 7.1111   *floating point data type* (*float*)

try to input the following commands in Python shell:

```
>>> type(23)

>>>type(1.23)

>>>type(2.0)

>>>my_number=23.1
>>>type(my_number)
```

The function type *returns us the type of the value.*

! float type stores only approximations to real numbers; there is a *limit to the precision*, or *accuracy*

3.2 Type conversions and rounding

There are situations where a value may need to be converted from one data type to another.

Try in the Python interpreter:

```
>>> int(4.5)
4
>>> float(3)
3.0
>>> float(int(4.5))
4.0
```

3.2 Type conversions and rounding

There are situations where a value may need to be converted from one data type to another.

Try in the Python interpreter:

```
>>> int(4.5)
4
>>> float(3)
3.0
>>> float(int(4.5))
4.0
```

note that the decimal part is simply cut off.

3.2 Type conversions and rounding

There are situations where a value may need to be converted from one data type to another.

Try in the Python interpreter:

```
>>> int(4.5)
4
>>> float(3)
3.0
>>> float(int(4.5))
4.0
>>> float("4.56")
4.56
>>> int("456")
456
```

note that the decimal part is simply cut off.

3.2 Type conversions and rounding

There are situations where a value may need to be converted from one data type to another.

Try in the Python interpreter:

```
>>> int(4.5)
4
>>> float(3)
3.0
>>> float(int(4.5))
4.0
>>> float("4.56")
4.56
>>> int("456")
456
```

note that the decimal part is simply cut off.

this is what we can use instead of eval at input

3.2 Type conversions and rounding

Consider the following code:

```
def main():
    x=float(input("Enter a decimal number:"))
    print("You entered: ",x)

    y = int(input("Enter an integer:"))
    print("You entered:",y)

main()
```

3.2 Type conversions and rounding

Consider the following code:

```
def main():
    x=float(input("Enter a decimal number:"))
    print("You entered: ",x)

    y = int(input("Enter an integer:"))
    print("You entered:",y)

main()
```

First run:

Enter a decimal number:5.6
You entered: 5.6
Enter an integer:8
You entered: 8

3.2 Type conversions and rounding

Second run:

```
Enter a decimal number:3.4
You entered:  3.4
Enter an integer:3.4
Traceback (most recent call last):
  File "/Users/luis/teaching/classes/22-1/csi31/webpage/luislectures/Lecture05/example.py", line 10, in <module>
    main()
  File "/Users/luis/teaching/classes/22-1/csi31/webpage/luislectures/Lecture05/example.py", line 7, in main
    y = int(input("Enter an integer:"))
ValueError: invalid literal for int() with base 10: '3.4'
```

3.2 Type conversions and rounding

In addition, *numeric type conversion* in place of `eval` does not accommodate simultaneous input.

```
>>> x,y=float(input("Enter two decimal values:"))
Enter two decimal values:5.6,7.5
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    x,y=float(input("Enter two decimal values:"))
ValueError: could not convert string to float:
'5.6,7.5'
```

This is a small price to pay for added security

3.2 Type conversions and rounding

To round off use `round` method.

```
round(number[,ndigits]) → number
```

Try the following in the Python interpreter:

```
>>> round(4.456,2)
4.46
>>> round(4.456,1)
4.5
>>> round(123.78476)
124
```

3.3 Using the Math Library

Python provides many other useful mathematical operations in a special *math library*

A *library* - is a module that contains some useful definitions of functions.

In order to use functions from the library we need to *include it* or *import it* to our program:

```
import math
```

Python	mathematics	English
pi	π	An approximation of pi
e	e	An approximation of e
sqrt(x)	$\sqrt{x}$	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 of x
acos(x)	arccos x	The inverse of cosine of x
atan(x)	arctan x	The inverse of tangent of x
log(x)	ln x	The natural logarithm of x
log10(x)	$\log_{10} x$	The common logarithm of x
exp(x)	e^x	The exponential of x
ceil(x)	$\lceil x \rceil$	Ceiling function of x
floor(x)	$\lfloor x \rfloor$	Floor function of x

3.3 Using the Math Library

see table 3.2 on page 68 for some math library functions.

see also Python Documentation -> The Python Standard Library -> Numeric and Mathematical Modules -> math

input the following statements in the Python Interpreter:

```
>>> import math
>>> math.sqrt(5)

>>> math.sqrt(25)
>>> math.ceil(234.345)
```

3.4 Accumulating Results: Factorial

factorial function: n!

n! = n(n-1)(n-2)(n-3)·...·3·2·1 = 1·2·3· ...·(n-3)(n-2)(n-1)n

Examples: 2! = 2·1=2 4! = 4·3·2·1 = 24

Let's write a program that calculates the factorial of a number entered by the user:

3.4 Accumulating Results: Factorial

factorial function: n!

n! = n(n-1)(n-2)(n-3)·...·3·2·1 = 1·2·3· ...·(n-3)(n-2)(n-1)n

Input: a positive integer (n)
Output: a positive integer (factorial)
Relationship: *factorial = n(n-1)(n-2)*...*2*1*

3.3 Accumulating Results: Factorial

factorial function: n!

n! = n(n-1)(n-2)(n-3)·...·3·2·1 = 1·2·3· ...·(n-3)(n-2)(n-1)n

Input: a positive integer (n)
Output: a positive integer (the factorial of n)
Relationship: *factorial = n(n-1)(n-2)*...*2*1*

Algorithm:
input number to take the factorial of, *n*
for loop that will iterate n times
    factorial = factorial * factor
output *factorial*