Getting Started

CSE 231, Rich Enbody

Office Hours

- After class
- By appointment – send an email
Project 1

- Python arithmetic
- Do with pencil, paper and calculator first
- Idle
- Handin
- Help room

What is a Computer Program?
Program

- A program is a sequence of instructions.
- To *run* a program is to:
  - create the sequence of instructions according to your design and the language rules
  - turn that program into the binary commands the processor understands
  - give the binary code to the OS, so it can give it to the processor
  - OS tells the processor to run the program
  - when finished (or it dies :-), OS cleans up.

Python is an *interpreted* language

- You can simply open the Python interpreter, and enter instructions one-at-a-time.

Or

- You can *import* a program which causes the instructions in the program to be executed, as if you had typed them in.
First Program: number_input.py

# input two numbers, add them together, print them out
num_str1 = input('Please enter an integer:')
num_str2 = input('Please enter a floating point number:')

str1_int = int(num_str1)
str2_float = float(num_str2)  # this is a comment

print('The numbers are: ', str1_int, ' and ', str2_float)
print('Their sum is: ', str1_int + str2_float, '
  ' and their product is: ', str1_int * str2_float)
input

The function:

```python
input("Enter a value")
```

- prints “Enter a value” in the Python shell and waits till the user types something (anything), followed by “Enter”
- Warning, it returns a string (sequence of characters), no matter what is given, even a number (‘1’ is not the same as 1, different types)

What's a string

The word "string" is used to indicate a sequence of characters, a compositor's term
print
my_var = 12
print(‘My var has a value of:’, my_var)

- `print` takes a list of elements to print, separated by commas
  - if the element is a string, prints it as is
  - if the element is a variable, prints the value associated with the variable
  - after printing, moves on to a new line of output

At the core of any language

- Control the flow of the program
- Construct and access data elements
- Operate on data elements
- Construct functions
- Construct classes
- Libraries and built-in classes
Save as a “module”

- Save with a .py suffix, so it becomes a Python module
- You “run” the module from the IDLE menu (F5) to see the results of the operation
- A module is just a file of Python commands

Errors
Common Error

- If you save the file without a .py suffix, it will stop colorizing and formatting the file.
- Fix: save with the .py

Syntax

- Lexical components.
- A Python program is:
  - A module (perhaps more than one)
  - Each module has Python statements
  - Each statement has expressions
Modules

http://pypi.python.org

Statements are commands in Python. They perform some action, often called a side effect, but they do not return any values
Expressions perform some operation and return a value.

Expressions can act as statements, but statements cannot act as expressions (more on this later).

Expressions typically do not modify values in the interpreter.

Note that MPL sometimes only asks for an expression.
What is the difference between a side effect and a return?

- \(1+2\) returns a value (it’s an expression). You can “catch” the return value (\(x=1+2\)). However, nothing else changed.
- \(\text{print(“hello”)}\) doesn’t return anything, but something else, the side effect, did happen. Something printed!

Python name conventions

- must begin with a letter or `_`
  - \(\text{Ab123}\) is OK, but \(\text{123ABC}\) is not.
- may contain letters, digits, and underscores
  - \(\text{this_is_an_identifier_123}\)
- may be of any length
- upper and lower case letters are different
  - \(\text{LengthOfRope}\) is not \(\text{lengthofrope}\)
- names starting with `_` have special meaning. Be careful
A comment begins with a “#”

- From the “#” to the end of that line: nothing will be interpreted by Python.
- Comments help the human reader.

Code as essay.
Knuth, *Literate Programming* (‘84)

Let us change our traditional attitude to the construction of programs:

Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

---

**Python types**

- integers (int): 5
- floating point (float): 1.2
- Booleans (bool): True
- Strings (str): “anything” or ‘something’
- lists: ['a',1,1.3]

Others…
A type in Python defines two things:
  o what it contains
  o the kinds of operations you can perform on it

Fundamental Types

- Integers (int)
  o 1, -27
- Reals (float)
  o 3.14, 10., .001, 3.14e-10, 0e0
- Booleans
  o True, False note the capital
Example 2

Worksheet 1
A character ‘1’ is not an integer 1.
- You need to convert the value returned by the `input` command (characters) into an integer (`int`)
- `int("123")` yields the integer 123

Type conversion
- `int(some_var)` converts to an integer
- `float(some_var)` converts to a float
- `str(some_var)` converts to a string
import math imports the math module

- Brings in Python statements to support math
- Use math prefix: math.xxx
  math.pi is π
  math.sqrt(2) finds the square root of 2

When = doesn’t mean equal

my_int = my_int + 7
= is assignment

\[ lhs = rhs \]

What “assignment” means is:
- evaluate all the “stuff” on the rhs of the =
- take the resulting value and associate it with the name on the lhs

Variable Objects
- Python maintains a list of pairs for every variable:
  - variable’s name
  - variable’s value
- A variable is created when a value is first assigned. It associates a name and a value.
- Subsequent assignments update the associated value.
- We say a name references a value.
- A variable’s type depends on what is assigned.

\[
\begin{array}{|c|c|}
\hline
\text{Name} & \text{Value} \\
\hline
\text{X} & 7 \\
\hline
\end{array}
\]
**Namespace:**
a table that associates a name with a value.

**Assignment Statement**

- Example: \( x = 2 + 3 \times 5 \)
  - evaluate expression \((2+3*5): 17\)
  - change the value of \( x \) to reference 17

- Example (\( y \) has value 2): \( y = y + 3 \)
  - evaluate expression \((y+3): 5\)
  - change the value of \( y \) to reference 5
x = 2 + 3 * 5
y = 2
y = y + 3

More on types

- Python does not require you to pre-define the type of a variable
- What type a variable holds can change
- Nonetheless, knowing the type can be important for using the correct operation on a variable.
x = 2
x = 7.5
x = 'abc'

What can go on the Lhs?

- The Lhs must indicate a name with which a value can be associated.
- Must follow the naming rules
  my_int = 5, Yes
  my_int + 5 = 7, No
More on whitespace

- For the most part, you can place “white space” (spaces) anywhere in your program
- Use it to make a program more readable

```
1 + 2 - 4
```
continuation

Python is sensitive to the end of line. To make a line continue, use: \
print("this is a test", \
"of continuation")
prints
this is a test of continuation

Python is sensitive to tabs. (indentation)
Syntax: tokens

Python Keywords: cannot use as variable names

and  del  from  not  while
as  elif  global  or  with
assert  else  if  pass  yield
break  except  import  print
class  exec  in  raise
continue  finally  is  return
def  for  lambda  try
Python Operators

+  -  *  **  /  //  
<<  >>  &  |  ^  ~
<  >  <=  =>  ==  !=
+=  -=  *=  /=  //=  %=  
&=  |=  ^=  >>=  <<=  * *=

Python Punctuators and Delimiters

($ and ? not allowed)

\  "  #  \n(  )  [  ]  {  }  @
,  :  .  `  =  ;
Operators

- Integer (int)
  - addition and subtraction: `+`, `-`
  - multiplication: `*`
  - division: 3 kinds!
    - division: `/`
    - quotient: `//`
    - remainder: `%`
  - exponent: `**`

- Floating point (float)
  - add, subtract, multiply, divide, exp: `+`, `-`, `*`, `/`, `**`

Types and division

```
\[ \frac{5}{3} \]
```

- `5/3` is division, yields float: 1.66666
- `5//3` is the quotient, yields integer: 1
- `5%3` is the remainder, yields integer: 2

```
\[ \frac{5.0}{3.0} \]
```

- `5.0/3.0` is division, yields float: 1.66666
Mixed Types

- \(4 \div 3\) is 1 (integer quotient)
- \(4.0 \div 3.0\) is 1.3333333 (float division)

What is \(4 \div 3.0\)?
- No mixed-type operations. Must convert.
- Python automatically converts to float.
  Thus \(4 \rightarrow 4.0\) so the result is 1.3333333

Try

- 5//2
- 5//2.0
- 3+4.0
- 1234//100
- 1234%100
- 1234//100%10
- int(1.234)
- float(2)
Collections of data types
(later in semester)

- list
  - sequence of any data elements
- dictionary
  - a collection of name:value pairs. Very powerful!
- class
  - a user-defined data type

Collections and building blocks
(examples of what’s to come later)

- Point (x,y,z are real numbers)
- Plane (3 points in a plane)
- Line segment (2 end points)
- Face (N points on a plane)
- Solid (N faces)
- Solid (N planes)
Note

- The interpreter translates Python into machine (computer) language. The first stage of that process is determining if it is valid Python code.
- If the program is not a valid Python program, the interpreter cannot translate it.
- If the interpreter cannot translate your code, it is not a Python program so it is not worth any points.