CISS450: Artificial Intelligence
Lecture 1: Overview

Yihsiang Liow

Agenda

- Examples
- Hands-on Quick Tour
- Introduction to Python

Examples

- Pre-requisite: I assume you already know C++ (CISS240 + CISS245)
- I'm using Python 3.7.9.
- The following are some simple programs
- There are two versions for each: The one on the left is written in C++, the one on the right is written in Python
- Compare the two and note the syntax differences
- For those without C++, just focus on the Python code

Examples

```
#include <iostream>

int main()
{
  std::cout
    << "Hello, World!\n";

  return 0;
}
```

```
print("Hello, World!")
```

Examples

```
#include <iostream>
#include <cmath>

int main()
{
 double princ = 1000;
 double rate = 0.85;
 double time = 3;

 std::cout
  << "Final principal:"
  << princ * pow(1+rate,3)
  << "\n";
}
```

```
princ = 1000.0
rate = 0.85
time = 3

print("Final principal",
    princ * (1 + rate)**3)
```

Examples

```
#include <iostream>

int main()
{
  int x = 5;
  if (x != 0)
  {
    std::cout <<"nonzero\n";
    X = 1;
  }
  else
  {
    std::cout << "zero\n";
    X = 0;
  }
  std::cout << x << '\n';

  return 0;
}
```

```
x = 5
if x != 0:
    print("nonzero")
    x = 1
else:
    print("zero")
    x = 0
print(x)
```

Examples

```
#include <iostream>
using std::cout;

int main()
{
  int s = 0;
  for (int i=1; i<=5; i++)
  {
    s += i;
    cout << i << ","
         << s << "\n";
  }
  std::cout << s << "\n";

  return 0;
}
```

```
s = 0
for i in [1, 2, 3, 4, 5]:
    s += i
    print( i, ",", s)
print(s)
```

IDLE

- Optional: Now move on to IDLE Tutorial …
- From 2012 on, a linux environment will be used.
- For the linux environment:
  - Interactive: Execute `python` at the command line prompt to invoke the interpreter. Type in python commands at the prompt. To exit interpreter, do Ctrl-D.
  - Non-interactive: Enter python commands into a text file `program.py`. Execute `python program.py` at the command line prompt.

Hands-on Quick Tour

- The next few slides will be a series of programs that you have to type in and run. I believe that when you first learn a language, you have to ***use*** it.
- So don't just stare at it even though it might obvious
- You will get a feel for some features of the language, do a simple GUI program, and write an over-simplified "search engine"
- We will cover the concepts slowly in the coming lectures

Program #1

- Type your first program in the Python shell window:

```
>>> print("Hello, World!")
Hello, World!
```

- Next, create a program called `hello.py` and run it

Program #2

- More on print:

```
>>> print     ("Hello, World!")
Hello, World!
>>> print \
        ("Hello, World!")
Hello, World!
>>> PRINT("Hello, World!")
```

Program #3

- In the shell, compute 2 to the power of 10, 100, 1000, 10000. Compare with C/C++.

```
>>> 2**10
1024
>>> 2**100
1267650600228229401496703205376
```

- Try:

```
>>> 5/3
1.6666666666666667
>>> 5 // 3
1
```

- What is PEMDAS? Try:

```
>>> 4 - (5 - 6) - 7 * 3 + 4 / 2 ** 2
```

Program #4

- Now for variables
- Use the editor, save, and run the following program. You don't have to create a new program, just use your `hello.py`:

```
s = "Hello, World!”
print("1.", s)
x = 2
y = 100
print("2.", x + y, x - y)
print("3.", x * y, x / y)
a = 3 % x
print("4.", a, x % 3)
print("%s. %s%s" % (5, a, x % 3))
```

Program #5

- Now for floats
- Try this

```
x = 1 / 4
y = 1.0 / 4
z = 1 / 4.0
print(x, y, z)
```

- Recall: In C/C++, there are several floating point types such as `double` and `float`. In Python there's only a float type which is the same as the `double` in C++.

Program #6

- Now for types
- Try:

```
x = 1
y = 1.0
z = "1"
print("1.", type(x))
print("2.", type(y))
print("3.", type(z))
```

- What does the `type` function do?

Program #7

- Now for inputs
- Try:

```
x = int(input("Enter an integer (say 1): "))
y = float(input("Enter a float (say 1.0): "))
z = input("Enter a string (say 1): ")
print("1.", x, type(x))
print("2.", y, type(y))
print("3.", z, type(z))
a = input("Enter a string (say abc): ")
```

Program #8

- Try:

```
print("ACT: Area Calculator for Triangles")
print("Copyright NASA")
base = int(input("Enter base: "))
height = int(input("Enter height: "))
print("Area =", base * height / 2)
```

- Try this out with base=0.5, height=12

Program #9

- For the C/C++ programmer, here's a surprise.
- Try:

```
x = 5
print("1.", x, type(x))
x = 6
print("2.", x, type(x))
x = "Hello, World!"
print("3.", x, type(x))
```

- What's the surprise?
- Also try:

```
x = 5
print("1.", isinstance(x, int))
print("1.", isinstance(x, str))
```

Program #10

- Try:

```
x = 5    # To be or not to be ...
```

- and then this:

```
x = 5      That is the question
```

- What's the point of #?

Program #11

- Python's version of "arrays", lists, have nice features. Try:

```
list = [3,1,6,23,-100,42,42,4,99,0]
print("1.", list, type(list))
x = list[0]
y = list[3]
print("2.", x, type(x), y, type(y))
print("3.", list[-1], list[-3])
z = list[2:5]
print("4.", z, type(z))
print("5.", list[2:], list[:5])
print("6.", list[:], list[2:-3])
print("7.", list[8:2:-1], list[:-3:-2])
```

Program #12

- More surprisingly nice features of lists:

```
list0 = ["primes:",  2, 3, 5, 7]
list1 = [11, 13, 17, 18]
list2 = list0 + list1
print("1.", list2, type(list2))
list2.remove(18)
print("2.", list2, type(list2))
list2.append(19)
print("3.", list2, type(list2))
print("4.", len(list2))
empty = []
print("5.", empty, type(empty))
```

Program #13

- A warning:

```
my_bank_acct = ["Yihsiang Liow", 100.00]
your_bank_acct = my_bank_acct
print(my_bank_acct, your_bank_acct)
my_bank_acct[1] =  110.50
print(my_bank_acct, your_bank_acct)
```

- Why? Let's have some sophisticated buzzwords ...

Program #14

- Break for some GUI fun. You might need to install Tkinter. “Hello, World!” on steriods:

```
from tkinter import Tk, Frame, Label, Button

window = Tk()

frame = Frame(window)
frame.pack()

label = Label(frame, text="hello world")
label.pack()

button = Button(frame, text="Exit",command=window.destroy)
button.pack()

window.mainloop()
```

Program #15

- Now for functions. Here are some:

```
def helloWorld():
    print("Hello, World!")

def hello(s):
    print("Hello,", s, "!")

helloWorld()
hello("Galaxy")
```

Program #16

- Another function:

```
def add(x, y):
    return x + y


def average(x, y, z):
    avg = (x + y + z) / 3.0
    return avg


print("1.", add(1, 2))
print("2.", add([2], [3,5]))
print("3.", add("Hello, ", "World!"))
```

- Create a function, `subtract`, that returns the difference of two arguments passed in. Test it.

Program #17

- Now for `if`. Try this:

```
s = input("Do not enter 42 ... ")
if s == "42":
    print("I said do NOT enter 42!!!")
```

and

```
s = input("Do not enter 42 ... ")
if s == "42":
    print("I said do NOT enter 42!!!")
else:
    print("Thanks")
```

Program #18

- And for `while`-loop, try

```
tries = 0
prompt = "Do not enter 42 ... "
s = input(prompt)
while s == "42":
    print("I said do NOT enter 42!!!")
    tries = tries + 1
    s = input(prompt)

if tries > 0:
    print("Can't you follow instructions??")
    print("You did that in",tries,"tries!!!")
else:
    print("Thanks")
```

Program #19

- And here's the `for`-loop. Try

```
for i in [5, 4, 3, 2, 1]:
    print(i, "...")
print("Blast off")
```

and

```
for i in [1, 2, 3, 4, 5]:
    if i % 2 == 1:
        print("she loves me ...")
    else:
        print("she loves me not ...")
print("Oops, I just killed a flower ...")
```

Program #20

- First, what does `find` do?

```
s = "Hello, World!"
print("1.", s.find("World"))
print("2.", s.find("world"))
print("3.", s.find("?"))
```

- Now try

```
import requests
s = requests.get("http://news.yahoo.com").text
print(s[:50])
print(s[-50:])
print(type(s))
print(s.find("Obama"))
print(s.find("Liow"))  # too bad
```

- Unicode type is similar to string type. Google "python unicode" for more information.

Program #21

- Try:

```
import requests

def match(url, keyword):
    s = requests.get(url).text
    return s.find(keyword)

word = input("What do you want to read? ")
url = "http://news.yahoo.com"
if match(url, word) >= 0:
    print(word, "found at", url)
else:
    print(word, "not found at", url)
```

Program #22

- Save this and run it:

```
def f(x):
    print(1 // x)
def g(x):
    f(x)
g(0)
```

- Error msg:

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in g
  File "<stdin>", line 2, in f
ZeroDivisionError: integer division or modulo by zero
```

Program #23

- Continuing above:

```
def f(x):
    print(1 // x)
def g(x):
    f(x)
try:
    g(0)
except ZeroDivisionError as e:
    print(“caught ZeroDivisionError”)
    print(“here’s the exception:”, e)
    print(“TOO BAD!”)
```

Program #24

- Continuing above:

```
def f(x):
    print(1 // x)
def g(x):
    f(x)
try:
    g(0)
except Exception as e:
    print(“caught some exception”)
    print(“here’s the exception:”, e)
    print(“TOO BAD!”)
```

- The DivisionError class is subclass of the Exception class.

Python Overview

- What is Python?
- Who is using Python?
- When should we not use Python?
- Resources

What is Python?

- Python is an object-oriented programming language invented by Guido van Rossum in 1987
- It's a general-purpose, open source object-oriented programming language optimized for quality, productivity, portability and integration
- The syntax is remarkable simple and clean - “Executable pseudocode”
- You can easily integrate with C
- There is a large collection of libraries
- “Pascal of the 2000s” - ideal for teaching because of simplicity

Who is using Python?

- **http://www.python.org/community/users.html**
- **http://www.pythonology.com/success**
- **Some big names:**
  - **Yahoo, Google, Infoseek, MCI**
  - **NASA, US Navy, US Dept of Agri.**
  - **Industrial Light & Magic, MayaVi Sci Data Viz**
  - **National Weather Service, Los Alamos NL, Lawrence Livermore NL**
  - **IBM, Philips semiconductor, Nokia**
- **10 famous websites written in python:**
  - Google, youtube, quora, dropbox, yahoo!, reddit, instagram, spotify, survey monkey, bitly

Resources

- The starting point is always `www.python.org`. This is your best friend. Get to know it well.
- Because Python is an open source product, there is a huge support base from its community. Check out `www.python.org` for communities, user groups, mailing lists
- There are also many tutorials, articles, free online books at `www.python.org`. The windows installation includes documents/manuals/tutorials.
- You can also find a list of recommended Python books at `www.python.org`

Objectives

- Besides the objectives in the syllabus, I hope to train you to look for info on your own. This is an absolutely crucial skill for survival in research and industry. However you cannot google for answers to assignments.
- ADVICE: At first this is confusing if you have only one other language (example: C/C++). But after a while you see the benefit of knowing several languages. You begin to see the similarities. And you see the underlying language constructs and algorithms better. So just keep using the language.

Exercise

- Redo all your C++ assignments using Python.
- Can't find your C++ assignments? Use any C++ textbook (yours or borrowed from the library.)