Lecture 1

Istvan Albert
Information about the course

• Course webpage:

http://www.personal.psu.edu/iua1/

• Office hours: MW: 2-3pm (iua1@psu.edu)

• Homework due Thursday the week after it was given out.
Rationale for this course

• Life sciences are becoming a data driven science

• Data is represented as text files in various formats

• Computational classes are usually focused on computer science, algorithms.

• We will focus on information processing
Lecture topics

15 weeks – two lectures per week

• computational foundations
• biological data formats
• statistical methods and visualization
• software tools

We will discuss multiple issues at once
• A main topic each week

• Practical examples that tie in with the topic

• Finishing with in class exercises depending on the subject
Project formats

• The best motivator is doing something useful for yourself

• Let’s solve some of your problems, you may form groups of 2 people.

• Send me a proposal on what type of data analysis would you like to see (deadline October 1st)

• We will try to tailor the course to cover those topics in the second half of the semester
1. We have microarray data and sequencing data for a certain organism

2. We’d like to correlate gene expression values with sequencing read coverage around certain genomic markers

Then you need to **bring data** – either already published ones - or a small percentage of unpublished data (ask permission from your advisor)
Expectations

You can **only learn** by doing

Spend 2-3 hours outside class each week:

- Explore behaviors
- Expand the scope of the study
- Try new solutions

*Time flies when you know what you are doing.*
Good News

There is no other domain of knowledge where simply

exploring, experimenting,

leads to immediate and dramatic improvement in one’s abilities!
Computation = Thought

• Computational approaches reflect the thought process

• When we learn informatics, we learn how to think in a way that is easy to translate into computation

• There is no magic – it is just like any other subject matter
Computers speak a different language

It is called a **programming** language.

But designed by humans!

No wonder there are thousands of programming languages – each reflecting someone’s ideal on how **things should really** work.
Common traits

A problem that can be solved in one particular programming language can be solved in any other computational language.

Are programming language all alike?
"A problem that can be solved in one particular programming language can be solved in any other computational language"

The actual solution may be substantially different

The cognitive overhead may be very different (although the brevity of one particular example is not proof of the overall characteristics)
Time to solution

Two things determine time to solution

- How long it takes to create this solution (human time)
- How long it takes that solution to execute (machine time)

Different languages make different tradeoffs

- High-level languages allow individuals to express their thoughts more quickly ...
- ... but the more abstract the language, the more slowly it runs
Food for thought

What makes computers so *inhuman*?

*They always do exactly what we tell them to do*

Often we’re not sure what we’ve actually said to the computer.
What is Python

• A simple (but not simplistic!) programming language

• Very popular in certain sciences

• Named after the Monty Python comedy group (not the snake)
What is Python for

• We will use it to interact with data:
  – Extract information
  – Filter datasets
  – Automate tasks
  – Create charts
  – Transform datasets for other bioinformatics tools
Not really about Python

• This course isn't really about Python
  – It's about solving data analysis problems

• But we have to write the examples in something
  – Might as well choose something useful
  – And it puts everyone on an equal footing

• We want to build skills that are transferable
Python fundamentals

Understand the basic building blocks:

1. Every object has a **type**

2. Every object has a **name**

Today’s main lesson in one sentence.

**When you have a problem, check the type and the name!**

1. Unassigned expressions are allowed but in practice we always name them
```python
age = 23
print age
print type(age)
```

```
23
[type 'int']
```
More types and names

```python
age=23
weight=100.0
prod = age * weight
print age, type(age)
print weight, type(weight)
print prod, type(prod)
```

Command Output:
```
23 <type 'int'>
100.0 <type 'float'>
2300.0 <type 'float'>
```
Unexpected problem. Integer division.

```python
age=23
weight=100.0
prod = age/100 * weight
print age/100, type(age)
print prod, type(prod)
```

Command Output:

```
0 <type 'int'>
0.0 <type 'float'>
```
Solution: use the right data type
This you’ll often see this error

```
greeting = "Hello World"

print greetings
```
Just about all data starts out as a string type!
Some objects already know how to do certain tasks

Python has introspection – we can ask it to tell us what an object can do

1. The function that does this is called `dir`

2. Moreover we can always find more information by invoking the `help`
More about string attributes

The attributes with underscores are called special methods. (not visible here due to large fonts)

These are the attributes of interest:

```python
1 greeting = "Hello World"
2 print dir(greeting)
```

Output:

```
'isalnum', 'isalpha', 'isdigit', 'islower',
'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'partition',
'replace', 'rfind', 'rindex', 'rjust',
rpartition', 'rsplit', 'rstrip', 'split',
splitlines', 'startswith', 'strip',
'swapcase', 'title', 'translate', 'upper',
'zfill'
```
Check for these attributes on the previous slide

greeting = "Hello World"
print greeting.lower()
print greeting.upper()
This does not work. Why?

```python
greeting = "Hello World"
print greeting.replace() 
```

Traceback (most recent call last):
  File "C:\cygwin\home\ialbert\sources\ialbert-web\ppt\week1\test.py", line 2, in <module>
    print greeting.replace()
  TypeError: replace() takes at least 2 arguments (0 given)
Find out more about the replace attribute

```python
greeting = "Hello World"
print help(greeting.replace)
```

```
Help on built-in function replace:

replace(...)  
S.replace (old, new[, count]) -> string

Return a copy of string S with all occurrences of substring old replaced by new. If the optional argument count is given, only the first count occurrences are replaced.
```
Now we know the correct use case

greeting = "Hello World!"
print greeting.replace('World', 'Universe')

Hello Universe!
Homework 1

• Repeat all examples (no need to turn them in)

• How many special methods does a string have?

• How many “non-special” methods does a string have?

• What does the “swapcase()” method do?

• Use the “title()” method in an example.
Lecture 2

Istvan Albert
Huck Institutes for the Life Sciences
Today’s topic are numbers

• Every data analysis task ends up manipulating numbers in some fashion

• There are some properties that we need to be aware of.

• Most of the time they work as you think they should. When they don’t … its very devious.
Revisit a previous example

If it leads to such subtle errors why is there a distinction at all?

Why can’t we have a single “number” type that always does the “right” thing?
We also have representation errors

Not all numbers are affected

Play with the number, find some that are exact and some that have inexact representation
The actual (internal) representation for 0.1

0.1000000000000000055511151231257827021181583404541015625

More information details at

http://docs.python.org/tutorial/floatingpoint.html

And many other resources.
Not required for the course, it is more for your personal enlightenment
Our data already has errors

- **Multiple orders** of magnitude larger than the representation error

- We just need to ensure not to fool ourselves into claiming a larger accuracy than we actually have.

There is one important rule to remember

**Never compare floating point numbers for equality!**
Integer comparison are exact

Use the ‘==’ as equality operator
Float comparison are not exact
How do we compare floats?

I know. It seems more work than it should be.
Other traps when using floats

• Care must be taken when combining numbers of widely different magnitudes!

• It is not absolute sizes, but relative ones:

  • 1 vs 1000 billion
  • 1 vs 1000 billionth fraction.
Short detour with Excel

Type the formula \( =A1+B1-B1 \) into the box.

What does it show?
Same operation with Python

```python
a = 1E6
b = 1E30

print a, b
print a + b - b
```

What does it show? See for yourself.
Long, looooong integers

```python
# you can make them so huge
# that it cannot finish computing
# in a reasonable amount of time
a = 894**3039
print type(a)
print a
```

```
<type 'long'>
1302325729433506319956185197765966581725227413
0027323993161442061478330026714900023662346604
5488704651700983107551228386304595546772260023
3740906839862262731140840762669756531224198699
6975018469006537926230248992206127422119365819
7615034478264326021170022666498092707936633510
2741340547016232643639894655498155239591162347
```
Floating point conclusions

- Don’t test equality for floats! Use the size of the difference instead.

- Don’t put widely different magnitude numbers into the same formula

- Small numbers with small ones, large ones with large ones.

- Most problems naturally lend themselves to such processing. Usually it is us who end up trying to be too smart and “save” some time.
Casting between types

```python
test.py (C:\cygwin\home\ialbert\sources\ialbert-web\ppt\week1) - Komodo Edit 5.1

1. a = "123"
2. print type(a), a
3. # cast to float
4. a = float(a)
5. print type(a), a

Command Output

'python -t C:\cygwin\home\ialbert\sources\ialbert-web\ppt\week1\test.py > &1' returned 0

<type 'str'> 123
<type 'float'> 123.0
```
Some casts may fail. This is good!
Homework. Help me cast this string to an integer!

What do I need to have in the code box to get the output below:

```
1
2    a = "123.0"
3    print type(a), a
4
5
6
7
8
```

```
<type 'str'> 123.0
<type 'int'> 123
```