Learning python

Sunday, May 5, 2019 10:30 AM

For this class, a Jupyter computing cell contains Python code. We will write in Python 3 (the default).

In general, this code looks a lot like code from any other computer language, with some important exceptions:

- No "{}" structure for code ("{}" structures data instead).
- Statement separators like ";" are unnecessary and discouraged.
- Indentation matters.

A quick tour of some examples:

```
# print the string "hi there".
print("hi there")
# set x to value 1
x = 1
# print something about x
if x < 2:
    print("x is less than 2")
    print("you win!")
else:
    print("x is greater than or equal to 2")
```

This prints

```
hi there
x is less than 2
you win!
```

Note a few things:

- "if" statements end with : and the next line indents.
- When the indentation ends, the conditional ends.
- Functions – e.g. `print` – are called by adding "(…)" and arguments.

The best way to learn a new language:

- Write small programs.
- Predict what they will do.
- Test your predictions.

Couch's laws of learning to program:

- **Programs haven't contained bugs since computers were made solid state.** A "bug" was a moth in a relay!
- Any bug in a program is actually **a problem with your understanding.** The program does exactly what you tell it to do.
- Thus, **correct your understanding**, before you correct the program.

The best way to learn to program is to **apply the scientific method**, to wit:

- Predict what will happen.
- Try an experiment.
- Correct your prediction and iterate.

Printing in python

- The first thing you should learn in any computer language is how to print output.

- In general,
  ```
  print(x)
  ```
  prints the value of x, where x can be anything, including a variable, an arithmetic expression, etc.
- The most basic form of printing is:
  ```
  print("This is a message to myself.")
  ```
  which prints a message to you.
  The syntax
  ```
  "This is a message to myself."
  ```
  is a *string*.
  It can also be written with single quotes as
  ```
  'This is a message to myself.'
  ```
  These two ways of writing it are *equivalent.*

Printing more complex messages

- There are many, many ways to print.
- One way that is particularly useful is to use *formatting*.
- e.g.,
  ```
  x = 'foo'
  print("the value of x is {}".format(x))
  ```
- This prints
  ```
  the value of x is foo
  ```
- How this works:
  - The `{}` in the first string is a placeholder for a value.
  - The `.format(x)` specifies the value x.
  - The value of x is *substituted* for `{}` in the string.
- Try the following:
  ```
  x = 1
  print("the value of x is {}".format(x))
  ```

```
x = 2.7
print("the value of x is {}".format(x))
x = 'yo'
print("the value of x is {}".format(x))
```

What happened in the previous example:

- Variables like x in Python are *polymorphic*. A variable can take any type of value.
- We substituted variables of type integer, floating point, and string, and the print statement was the same for each.

Printing is your secret weapon:

- In Jupyter, when you write Python print statements into a cell and execute it, the `print` output is written at the end of the cell.
- Thus, you can learn to understand difficult things by using `print` to unravel what is happening.
- Even an advanced programmer (with 40+ years of experience, like myself) has to resort to printing to understand subtle programming problems.