Getting Started With Python Programming

How are computer programs created

Variables and constants

Input and output

Operators

Common programming errors

Advanced techniques for formatting text output

Reminder: About The Course Textbook

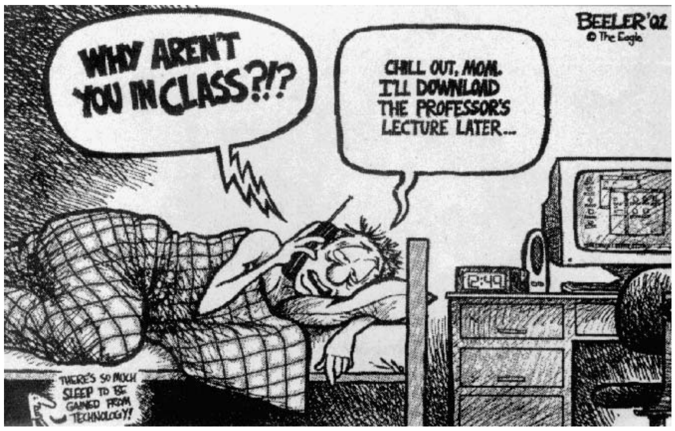
•It's recommended but not a required purchase.

•However the course notes are required for this course

Reminder: How To Use The Course Resources

•They are provided to support and supplement this class.

•Neither the course notes nor the text book are meant as a substitute for regular attendance to lecture and the tutorials.


BEELER'02
© The Eagle
WHY AREN'T YOU IN CLASS?!?
CHILL OUT, MOM. I'LL DOWNLOAD THE PROFESSOR'S LECTURE LATER...
12:40
THERE'S SO MUCH SLEEP TO BE GAINED FROM TECHNOLOGY!

Reminder: How To Use The Course Resources (2)

```
procedure add (var head        : NodePointer;
               var newNode : NodePointer);
var
  temp : NodePointer;
begin
  if (head = NIL) then
    head := newNode
  else
  begin
    temp := head;
    while (temp^.next <> NIL) do
      temp := temp^.next;
    temp^.next := newNode;
  end;
  newNode^.next := NIL;
end;
```

Reminder: How To Use The Course Resources (2)

```
procedure add (var head       : NodePointer;
               var newNode : NodePointer);
var
  temp : NodePointer;
begin
  if (head = NIL) then
    head := newNode
  else
  begin
    temp := head;
    while (temp^.next <> NIL) do
      temp := temp^.next;
    temp^.next := newNode;
  end;
  newNode^.next := NIL;
end;
```

If you miss a class make sure that you catch up on what you missed (get someone's class notes)

...when you do make it to class make sure that you supplement the slides with your own notes (cause you aint gonna remember it in the exams if you don't)

But Once You've Made An Attempt To Catch Up

- Ask for help if you need it
- There are no dumb questions

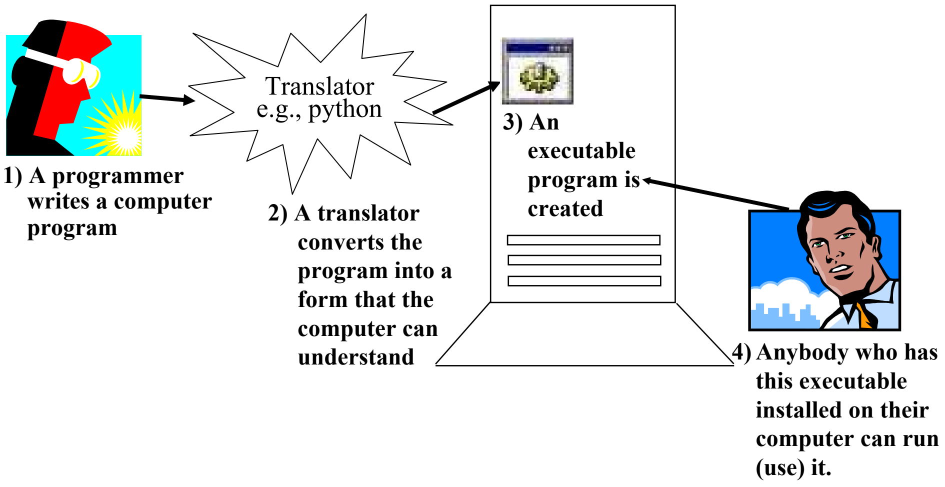
Image from "The Simpsons" © Fox

Don’t Forget: How To Succeed In This Course

1. Practice things yourself
2. Make sure that you keep up with the material
3. Look at the material before coming to lecture
4. Start working on things early

Computer Programs

Binary is the language of the computer


Translator
e.g., python
1) A programmer writes a computer program
2) A translator converts the program into a form that the computer can understand
3) An executable program is created
4) Anybody who has this executable installed on their computer can run (use) it.

Translators

Convert computer programs to machine language

Types

1) Interpreters
   - Each time that the program is run the interpreter translates the program (translating a part at a time).
   - If there are any errors during the process of interpreting the program, the program will stop running right when the error is encountered.
2) Compilers
   - Before the program is run the compiler translates the program (compiling it all at once).
   - If there are *any errors* during the compilation process, no machine language executable will be produced.
   - If there are *no errors* during compilation then the translated machine language program can be run.

Python

This is the name of the programming language that will be used to illustrate programming concepts this semester:

- My examples will be written in Python
- Your assignments will be written in Python

Some advantages:

- Free
- Powerful
- Widely used (Google, NASA, Yahoo, Activision, Electronic Arts etc.)

Named after a British comedy

Monty Python © BBC

Online documentation: http://www.python.org/doc/

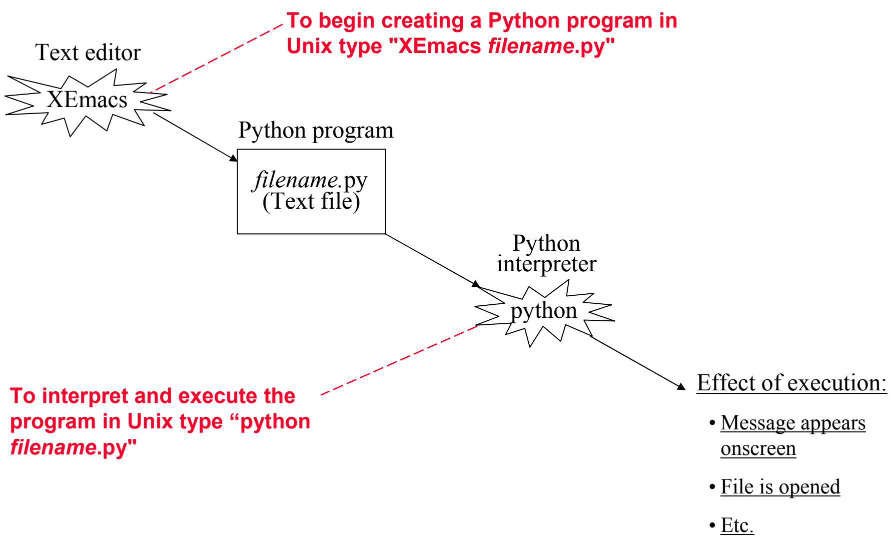
An Example Python Program

You can find an online version of this program in the UNIX file system under /home/courses/217/examples/intro/small.py:

Filename: small.py

```
print "hello"
```

Creating, Translating And Executing Python Programs


To begin creating a Python program in Unix type "XEmacs *filename*.py"
Text editor
XEmacs
Python program
filename.py
(Text file)
Python interpreter
python
To interpret and execute the program in Unix type "python *filename*.py"
Effect of execution:
• Message appears onscreen
• File is opened
• Etc.

Displaying String Output

String output: A message appears onscreen that consists of a series of text characters.

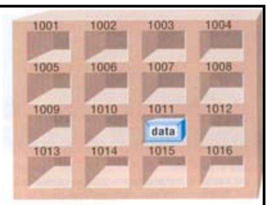
Format:

print "*the message that you wish to appear*"

Example:

```
print "foo"
print "bar"
```

Variables



Set aside a location in memory

Used to store information (temporary)

- This location can store one 'piece' of information
- *At most* the information will be accessible as long as the program runs

Some of the types of information which can be stored in variables:

- Integer
- Real numbers
- Strings

Picture from Computers in your future by Pfaffenberger B

Variable Naming Conventions

- Should be meaningful.
- Names *must* start with a letter (Python requirement) and *should not* begin with an underscore (style requirement).
- Can't be a reserved word or a keyword.
- Names are case sensitive but avoid distinguishing variable names only by case (bad style).
- Variable names should generally be all lower case.
- For variable names composed of multiple words separate each word by capitalizing the first letter of each word (save for the first word) or by using an underscore. (Be consistent!)

Constants

- Memory locations that shouldn't change.

- Used to make the program easier to read and understand:
  PI = 3.14

- Differentiated from variables by capitalization:
  - Multi-word constants can use the underscore to separate words e.g., MAX_SIZE = 10

Displaying The Contents Of Variables And Constants

Format:

```
print <variable name>
print <constant name>
```

Example:

```
aNum = 10
aCONSTANT = 10
print aNum
print aCONSTANT
```

Mixed Output

Mixed output: getting string output and the contents of variables (or constants) to appear together.

Format:

```
print "string", <variable or constant>
```

Examples:

```
myInteger = 10
myReal = 10.5
myString = "hello"

print "MyInteger:" , myInteger
print "MyReal:" , myReal
print "MyString:" , myString
```

Arithmetic Operators

Operator	Description	Example
=	Assignment	num = 7
+	Addition	num = 2 + 2
-	Subtraction	num = 6 - 4
*	Multiplication	num = 5 * 4
/	Division	num = 25 / 5
%	Modulo	num = 8 % 3
**	Exponent	num = 9 ** 2

Augmented Assignment Operators (Shortcuts)

Operator	Long example	Augmented Shortcut
+=	num = num + 1	num *= 1
-=	num = num – 1	num -= 1
*=	num = num * 2	num *= 2
/=	num = num / 2	num /= 2
%=	num = num % 2	num %= 2
**=	num = num ** 2	num **= 2

Program Documentation

Program documentation: Used to provide information about a computer program to another *programmer*:

- Often written inside the same file as the computer program (when you see the computer program you can see the documentation).
- The purpose is to help other programmers understand how the program code was written: how it works, what are some of it's limitations etc.

User manual: Used to provide information about how to use a program to *users* of that program:

- User manuals are traditionally printed on paper but may also be electronic but in the latter case the user manual typically takes the form of electronic help that can be accessed as the program is run.
- The purpose is to help users of the program use the different features of the program without mention of technical details.

Program Documentation (2)

- It doesn't get translated into binary.
- It doesn't contain instructions for the computer to execute.
- It is for the reader of the program:
  - What does the program do e.g., tax program.
  - What are it's capabilities e.g., it calculates personal or small business tax.
  - What are it's limitations e.g., it only follows Canadian tax laws and cannot be used in the US.
  - What is the version of the program
    - If you don't use numbers for the different versions of your program then consider using dates.
  - How does the program work.
    - This is often a description in English (or another high-level) language that describes the way in which the program fulfills its functions.
    - The purpose of this description is to help the reader quickly understand how the program works

Program Documentation (3)

Format:

# <Documentation>

$\underbrace{\phantom{\# <Documentation>}}$

The number sign '#' flags the translator that what's on this line is documentation.

Examples:

```
# Tax-It v1.0: This program will electronically calculate your tax return.
# This program will only allow you to complete a Canadian tax return.
```

Input

The computer program getting information from the user

Format:

<variable name> = input()
OR
<variable name> = input("*<Prompting message>*")

Example:

```
print "Type in a number: ",
num = input ()
        OR
num = input ("Type in a number: ")
```

Types Of Programming Errors

1. Syntax/translation errors
2. Runtime errors
3. Logic errors

1. Syntax/ Translation Errors

Each language has rules about how statements are to be structured.

English sentence is structured by the grammar of the English language:

- The cat sleeps the sofa.

Grammatically incorrect: missing the preposition to introduce the prepositional phrase 'the sofa'

Python statements are structured by the syntax of Python:

- $\underbrace{5 =}$ num

Syntactically incorrect: the left hand side of an assignment statement cannot be a literal constant.

1. Syntax/ Translation Errors (2)

The translator checks for these errors when a computer program is translated to binary:

- For compiled programs (e.g., C, C++, Pascal) translation occurs before the program is executed (because compilation occurs all at once before execution).
- For interpreted programs (e.g., Python) translation occurs as each statement in the program is executing (because interpreting occurs just before each statement executes).

1. Some Common Syntax Errors

Miss-spelling names of keywords

- e.g., ‘primt’ instead of ‘print’

Forgetting to match closing quotes or brackets to opening quotes or brackets.

Using variables before they’ve been named (allocated in memory). You can find an online version of this program in the UNIX file system under /home/courses/217/examples/intro/syntax.py:

```
print num
```

2. Runtime Errors

Occur as a program is executing (running).

The syntax of the language has not been violated (each statement follows the rules/syntax).

During execution a serious error is encountered that causes the execution (running) of the program to cease.

A common example of a runtime error is a division by zero error.

James Tam

2. Runtime Error: An Example

You can find an online version of this program in the UNIX file system under /home/courses/217/examples/intro/runtime.py:

```
num2 = input("Type in a number: ")
num3 = input("Type in a number: ")
num1 = num2 / num3
print num1
```

James Tam

3. Logic Errors

The program has no syntax errors.

The program runs from beginning to end with no runtime errors.

But the logic of the program is incorrect (it doesn't do what it's supposed to and may produce an incorrect result).

You can find an online version of this program in the UNIX file system under /home/courses/217/examples/intro/logic.py:

```
print "This program will calculate the area of a rectangle"
length = input("Enter the length: ")
width = input("Enter the width: ")
area = length + width
print "Area: ", area
```

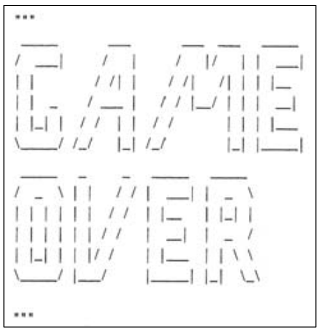
Advanced Text Formatting

•Triple quoted output

•Using escape sequences

Triple Quoted Output

- Used to format text output
- The way in which the text is typed into the program is exactly the way in which the text will appear onscreen.
- You can find an online example of triple quoted output in the UNIX file system under /home/courses/217/examples/intro/formatting1.py:



```
"""
 _____       ___       ___  ___   _____
/  ___|     /   |     /   |/   | |  ___|
| |        / /| |    / /|   /| | | |__
| |  _    / ___ |   / / |__/ | | |  __|
| |_| |  / /  | |  / /       | | | |___
\_____/ /_/   |_| /_/        |_| |_____|

 _____   _     _   _____   _____
/  _  \ | |   / / |  ___| |  _  \
| | | | | |  / /  | |__   | |_| |
| | | | | | / /   |  __|  |  _  /
| |_| | | |/ /    | |___  | | \ \
\_____/ |___/     |_____| |_|  \_\
"""
```

From Python Programming (2nd Edition) by Michael Dawson

Escape Codes

The back-slash character enclosed within quotes won't be displayed but instead indicates that a formatting (escape) code will follow the slash:

Escape sequence	Description
\a	Alarm. Causes the program to beep.
\b	Backspace. Moves the cursor back one space.
\n	Newline. Moves the cursor to beginning of the next line.
\t	Tab. Moves the cursor forward one tab stop.
\'	Single quote. Prints a single quote.
\"	Double quote. Prints a double quote.
\\	Backslash. Prints one backslash.

Escape Codes (2)

You can find an online version of this program in the UNIX file system under /home/courses/217/examples/intro/formatting2.py:

```
print "\a*Beep!*"
print "h\bello"
print "hi\nthere"
print 'it\'s'
print "he\\y \"you\" "
```

You Should Now Know

What is the difference between the two types of translators: compilers and interpreters.

How to create, translate and run Python programs on the Computer Science network.

Variables:

- What are they and what are they used for
- How to access and change the value of a variable
- Conventions for naming variables

Constants:

- What are constants and how do they differ from variables
- What are the benefits of using a named constant

You Should Now Know (2)

How are common mathematical operations performed in Pascal.

Output:

- How to display text messages or the value of a memory location (variable or constant) onscreen with print

Input:

- How to get a program to acquire and store information from the user of the program

What are the three programming errors, when do they occur and what is the difference between each one.

You Should Now Know (3)

How triple quotes can be used in the formatting of output.

What is an escape code and how they can affect the output or execution of a program.