Getting Started With Python Programming

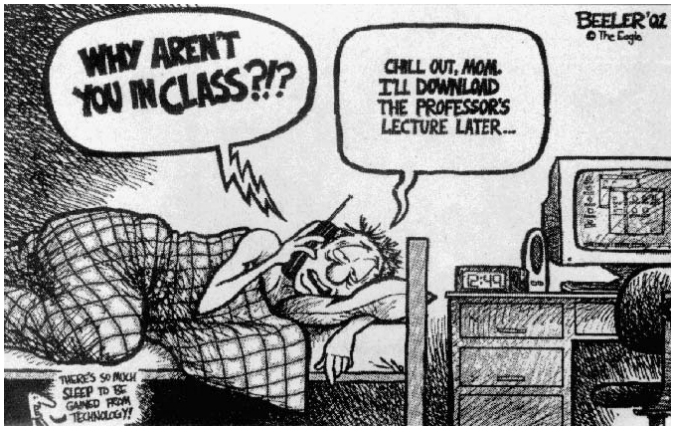
- How are computer programs created
- Representing information with binary
- Variables and constants
- Input and output
- Operators
- Common programming errors
- Advanced techniques for formatting text output

Reminder: About The Course Textbooks

- They're a recommended but not required.
- 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.



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^.next := newNode
  else
    begin
      temp := head;
      while (temp^.next <> NIL) do
        temp := temp^.next;
      temp^.next := newNode;
    end;
  newNode^.next := NIL;
end;
```

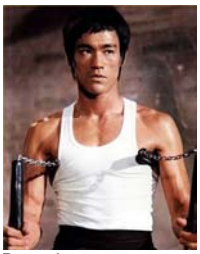
If you miss a class make sure that you catch up on what you missed (get someone's class notes)

...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)

How To Succeed

- Successful people


Leonardo da Vinci


Bruce Lee


J.R.R. Tolkien


Amadeus Mozart


Wayne Gretzky

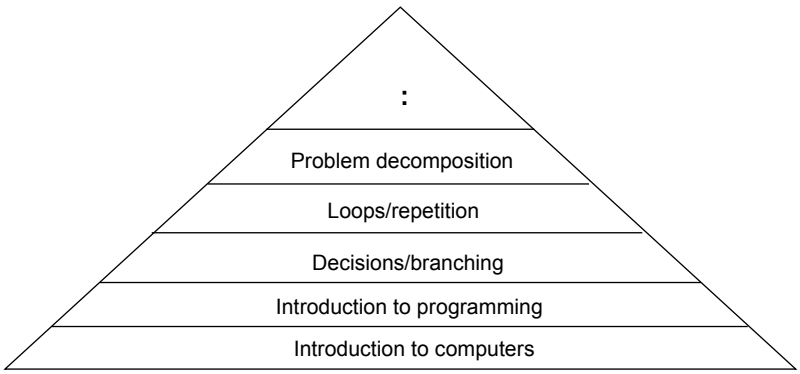
How To Succeed In This Course

1. Practice things yourself.
   - You get better by doing things for yourself (this is a 'hands-on' field of study and work).
   - Write lots programs.
     - At the *very least* attempt every assignment.
     - Try to do some additional practice work (some examples will be given in class, some practice assignments will be available on the course web page).
     - Write lots of little 'test' programs to help you understand and apply the concepts being taught.
   - Trace lots of code
     - Reading through programs that other people have written and understanding how and why it works the way that it does.

James Tam

How To Succeed In This Course (2)

2. Make sure that you keep up with the material
   - Many of the concepts taught later depend upon your knowledge of earlier concepts.
   - Don't let yourself fall behind!
   - *At least* attempt all assignments!



James Tam

How To Succeed In This Course (3)

3. Look at the material before coming to lecture so you have a rough idea of what I will be talking about that day:
    a) Read the slides
    b) Look through the textbooks (if you got it)

How To Succeed In This Course (4)

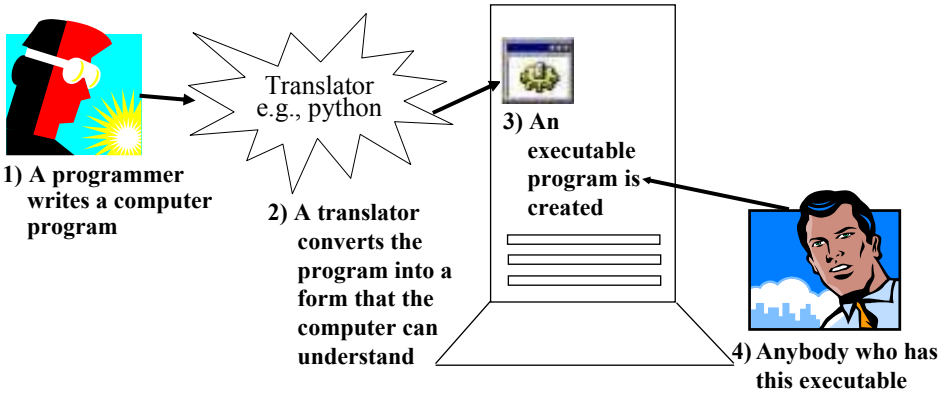
4. Start working on things as early as possible:
    - Don't cram the material just before the exam, instead you should be studying the concepts as you learn them throughout the term.
    - Don’t start assignments the night (or day!) that they are due, they may take more time than you might first think so start as soon as possible.

How To Succeed In This Course: A Summary

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



What Is Binary?

- (What you know): Binary is a method of representing information that uses two states.
- (What you may not be aware of): The number system that you are familiar (decimal) uses 10 states to represent information.

James Tam

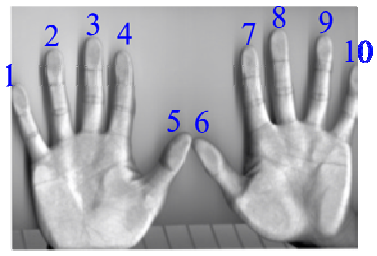
How Is Decimal Used To Store Numeric Information

- Base 10
  - 10 unique symbols are used to represent values

0
1
2
3
4
5
6
7
8
9
10
:

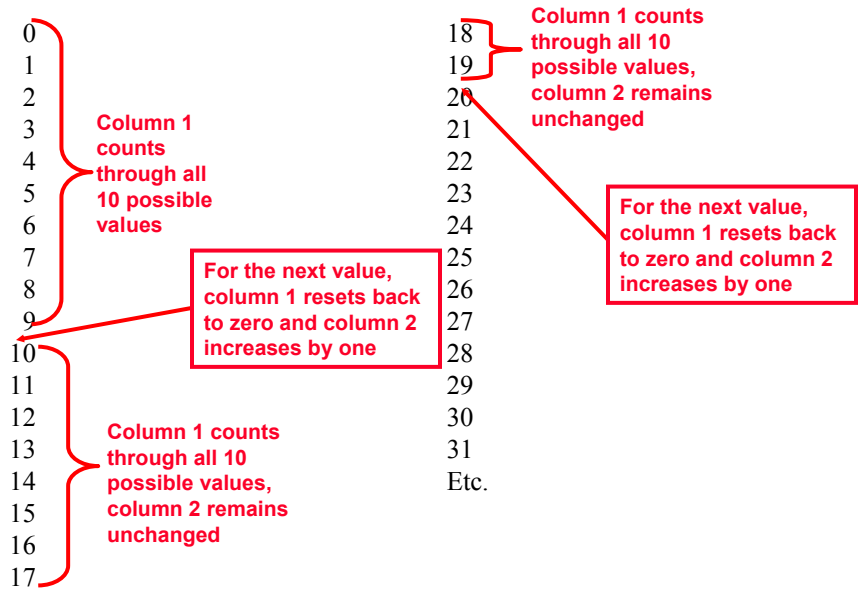
The largest decimal value that can be represented by a single decimal digit is 9 = base(10) - 1

The number of digits is based on…the number of digits



James Tam

How Does Counting In Decimal Work?



Decimal: Summary

- Base ten
- Employs ten unique symbols (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- Each digit can only take on the value from 0 – 9
  - Once a column has traversed all ten values then that column resets back to zero (as does it's right hand neighbours) and the column to it's immediate left increases by one.

Binary: Summary

- Base two
- Employs two unique symbols (0 and 1)
- Each digit can only take on the value 0 or the value 1
  - Once a column has traversed both values then that column resets back to zero (as does it's right hand neighbours) and the column to it's immediate left increases by one.

James Tam

Counting In Binary

Decimal value	Binary value	Decimal value	Binary value
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

James Tam

How Is Binary Used?

- Representing instructions in a computer program
  - E.g., print "End program." onscreen translates to
  - 0001 1000 1010 1111
  - 0100 0101
  - 0110 1110
  - Etc.
- Representing the data in a computer program
  - Alphanumeric text is represented with a numeric code (decimal ASCII value) which is translated to a binary ASCII value (e.g., 'A' = 65 = 0100 000). (In UNIX type "man ascii" for all the ASCII codes).
  - Signed integers can be represented by using one bit representing the *sign* of the number and the remainder of the bits representing the *size* of the number.

Sign bit:	**Remaining bits (n-1):**
0 = positive 1 = negative	Used to represent the size of the number.

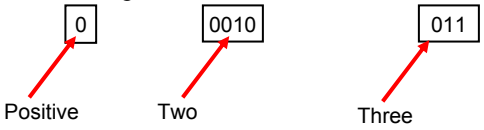
James Tam

How Is Binary Used? (2)

- Signed real numbers are more complex and require three parts

Sign bit	**Mantissa bits**	**Exponent bits**
b	bbbb	bbb

- The mantissa is raised to the exponent in order to determine the size of the number.
- Example:



Number: Positive 2 cubed...the number positive 8 is being represented in signed binary.

James Tam

Why Is It Important To Know How Data Is Being Stored?

- Different representations store different types of information but some have drawbacks.
- Real number representations may result in a loss of accuracy:
  - Only an approximation of some fractional values may be stored e.g., 1/3
  - Even storing some non-repeating fractional values may result in the loss of some information.
  - Example: suppose 1 digit is used for the sign, 5 for the mantissa and 3 for the exponent.
    - 123.45 is represented as $12345 * 10^{-2}$
    - 0.12 is represented as $12000 * 10^{-5}$
    - 123456 is represented as $12345 * 10^{1}$
- Morale of the story: Don't store information as a real number if it can be stored as an integer because of the potential loss of accuracy (e.g., store monetary values as cents rather than dollars).

Storing Other Information

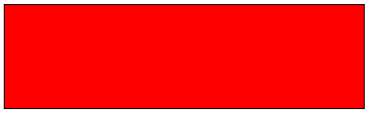
- Text: ASCII represents simple alphanumeric information

8 bits:
1 used for error checking
7 for the alphanumeric information = 128 combinations

- Text: beyond simple English representations
  - Arabic, Dutch, Chinese, French, German etc.
  - Representing this expanded text information uses additional bits:
    - 16 bits = 65,536 combinations
    - 24 bits = 16,777,216 combinations

Storing Other Information (2)

- Colors: using ~16 million colors can present a 'true life' representation, how are the color combinations encoded?



Converting From Binary To Decimal

- Start with some binary number to convert:
  - E.g., 1 0 1. 1
- Label each of the binary digits:
  - Starting with the digit immediately left of the decimal point and moving left (away from the decimal point) label the binary digits 0, 1, 2, 3 etc. in succession.
  - Starting with the digit immediately right of the decimal point and moving right (away from the decimal) label the binary digits -1, -2, -3...

    2 1 0 -1 ← **Position of each binary digit**

  - E.g., 1 0 1. 1 ← **Binary number to be converted**
- Evaluate the expression: the binary digit raised to some exponent[1], multiply the resulting expression by the corresponding digit and sum the resulting products.

  Value in decimal = $(1x2^2) + (0x2^1) + (1x2^0) + (1x2^{-1}) = (1x4) + (0x2) + (1x1) + (1 * 1/2) = 4 + 0 + 1 + 0.5 = 5.5$

1 The value of this exponent will be determined by the position of the digit (value of the superscript)

Binary To Decimal: Other Examples

- $0101.11_2 = ????_{10}$
- $100000_2 = ????_{10}$
- $011111_2 = ????_{10}$

Decimal To Binary

Split up the integer and the fractional portions:

1) For the integer portion:
   a. Divide the integer portion of the decimal number by two.
   b. The remainder becomes the first integer digit of the number (immediately left of the decimal) in binary.
   c. The quotient becomes the new integer value.
   d. Divide the new integer value by two.
   e. The new remainder becomes the second integer digit of the binary number (second digit to the left of the decimal).
   f. Continue dividing until the quotient is less than two and this quotient becomes the last integer digit of the binary number.

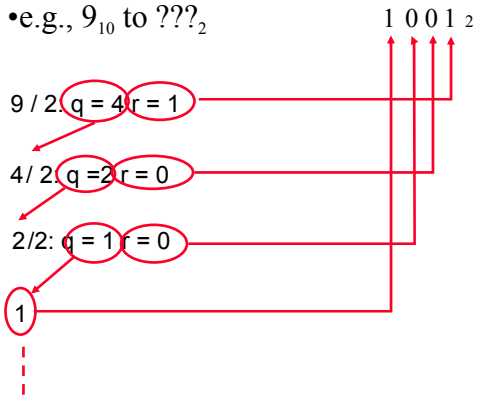
Decimal To Any Base (2)

2) For the fractional portion:

a. Multiply by two.
b. The integer portion (if any) of the product becomes the first rational digit of the converted number (first digit to the right of the decimal).
c. The non-rational portion of the product is then multiplied by two.
d. The integer portion (if any) of the new product becomes the second rational digit of the converted number (second digit to the right of the decimal).
e. Keep multiplying by two until either the resulting fractional part of the product equals zero or you have the desired number of places of precision.

Decimal To Any Base (2)

•e.g., 9_{10} to $???_2$



Stop dividing! (quotient less than target base)

Decimal To Binary: Other Examples

- $5.75_{10} = ????_2$
- $32_{10} = ????_2$
- $31_{10} = ????_2$

Python

- This is the name of the programming language that will be used to illustrate different 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/2.5.2/

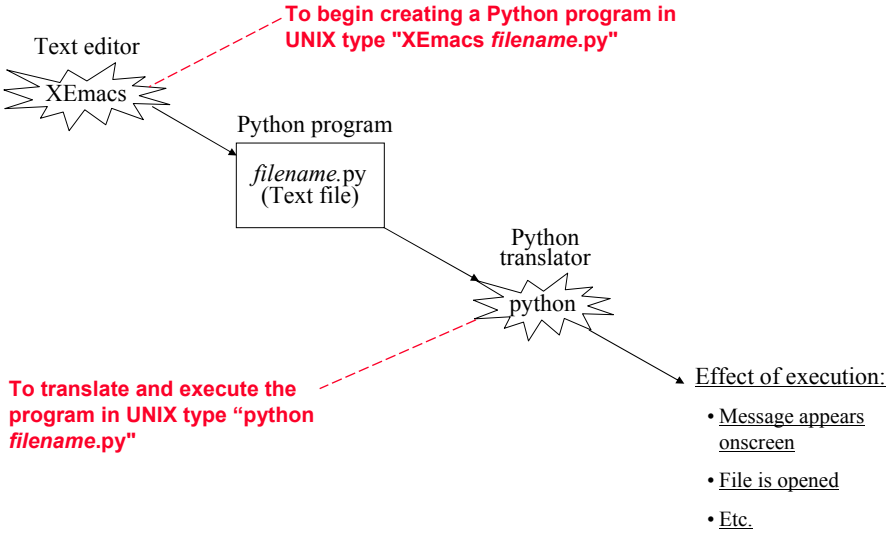
An Example Python Program

•You can find an online version of this program in UNIX under /home/231/examples/intro/small.py:

Filename: small.py

```
print “hello”
```

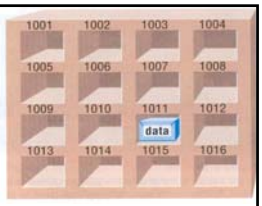

Creating, Translating And Executing Python Programs



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 keyword (see next slide).
- 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!)

Key Words In Python[1]

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

1 From "*Starting out with Python*" by Tony Gaddis

Constants

- Memory locations that shouldn't change.
- The naming conventions for choosing variable names generally apply to constants but the name of constants should be all UPPER CASE. (You can separate multiple words with an underscore).
- They are capitalized so the reader of the program can distinguish them from variables.
  - For some programming languages the translator will enforce the immutability of the constant.
  - For languages such as Python it is up to the programmer to recognize a constant for what it is and not to change it.

Why Use Constants

1. They make your program easier to read and understand

```
populationChange = (0.1758 – 0.1257) * currentPopulation;
```

Vs.

Magic Numbers (avoid whenever possible!)

```
BIRTH_RATE = 17.58
MORTALITY_RATE = 0.1257
currentPopulation = 1000000
populationChange = (BIRTH_RATE - MORTALITY_RATE) *
    currentPopulation
```

Purpose Of Named Constants (2)

- 2) Makes the program easier to maintain
  - If the constant is referred to several times throughout the program, changing the value of the constant once will change it throughout the program.

Purpose Of Named Constants (3)

```
BIRTH_RATE = 0.1758
MORTALITY_RATE = 0.1257
populationChange = 0
currentPopulation = 1000000
populationChange = (BIRTH_RATE - MORTALITY_RATE) * currentPopulation
if (populationChange > 0):
    print "Increase"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE, " Population
  change:", populationChange
elif (populationChange < 0):
    print "Decrease"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE,  "Population
  change:", populationChange
else:
    print "No change"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE,  "Population
  change:", populationChange
```

Purpose Of Named Constants (3)

One change in the initialization of the constant changes every reference to that constant

```
BIRTH_RATE = 0.8
MORTALITY_RATE = 0.1257
populationChange = 0
currentPopulation = 1000000
populationChange = (BIRTH_RATE - MORTALITY_RATE) * currentPopulation
if (populationChange > 0):
    print "Increase"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE, " Population
  change:", populationChange
elif (populationChange < 0):
    print "Decrease"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE,  "Population
  change:", populationChange
else:
    print "No change"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE,  "Population
  change:", populationChange
```


Purpose Of Named Constants (4)

One change in the initialization of the constant changes every reference to that constant

```
BIRTH_RATE = 0.1758
MORTALITY_RATE = 0.01
populationChange = 0
currentPopulation = 1000000
populationChange = (BIRTH_RATE - MORTALITY_RATE) * currentPopulation
if (populationChange > 0):
    print "Increase"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE, " Population
  change:", populationChange
elif (populationChange < 0):
    print "Decrease"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE,  "Population
  change:", populationChange
else:
    print "No change"
    print "Birth rate:", BIRTH_RATE, " Mortality rate:", MORTALITY_RATE,  "Population
  change:", populationChange
```

Named Constants And Python

- Using named constants is regarded as "good" style when writing a computer program.
- Some programming languages have a mechanism for ensuring that named constants do not change.
- Example:

```
A_CONSTANT = 100
A_CONSTANT = 12
```

With programming languages that enforce the immutability of constants this would result in an error

- Python does not enforce the immutability of constants so it is up to the person writing/modifying the program to avoid changing the value stored in a constant.

Displaying The Contents Of Variables And Constants

- **Format:**

print <*variable name*>
print <*constant name*>

- **Example:**

```
aNum = 10
A_CONSTANT = 10
print aNum
print A_CONSTANT
```

Mixed Output

- Mixed output: getting string output and the contents of variables (or constants) to appear together.
- **Format:**
  print "*string*", <*variable or constant*>, "*string*", <*variable or constant*> etc.

- **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

James Tam

Order Of Operation

- First level of precedence: top to bottom
- Second level of precedence
  - If there are multiple operations that are on the same level then precedence goes from left to right.

()	Brackets (inner before outer)
**	Exponent
*, /, %	Multiplication, division, modulo
+, -	Addition, subtraction

James Tam

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. In Canada it doesn't calculate taxes for organizations with a yearly gross earnings over $1 billion.
  - What is the version of the program
    - If you don't use numbers for the different versions of your program then consider using dates (tie this with program features).
  - 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.
    - Typically used to describe things that are not immediately self evident from the program code.

Program Documentation (3)

- **Format:**

  # <*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:**

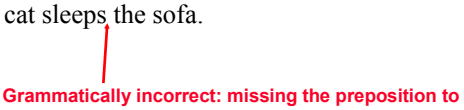
```
print "Type in a number: "
num = input ()
        OR
num = input ("Type in a number: ")
```

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.
- An 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:
  - 5 = num

Syntactically incorrect: the left hand side of an assignment statement cannot be a literal (unnamed) 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 once 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 UNIX under /home/231/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.
- With a language like Python where translation occurs just before execution the timing of when runtime errors appear won't seem different from a syntax error.
- But for languages where translation occurs well before execution the difference will be quite noticeable.
- A common example of a runtime error is a division by zero error.

2. Runtime Error: An Example

- You can find an online version of this program in UNIX under /home/231/examples/intro/runtime.py:

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

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 UNIX under /home/231/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 UNIX under /home/231/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 UNIX under /home/231/examples/intro/formatting2.py:

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


After This Section You Should Now Know

- The binary number system
  - How to count in binary
  - Conversions to/from binary
  - The different ways in which information is represented using the binary system
- How to create, translate and run Python programs on the Computer Science network.
- Variables:
  - What they are used for
  - How to access and change the value of a variable
  - Conventions for naming variables
- Named constants:
  - What are named constants and how do they differ from variables
  - What are the benefits of using a named constant

After This Section You Should Now Know (2)

- What is program documentation and what are some common things that are included in program documentation
- How are common mathematical operations performed
- Output:
  - How to display messages that are a constant string 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.
- 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.