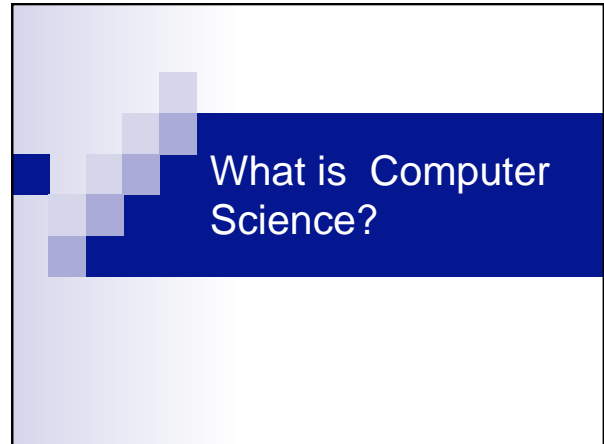
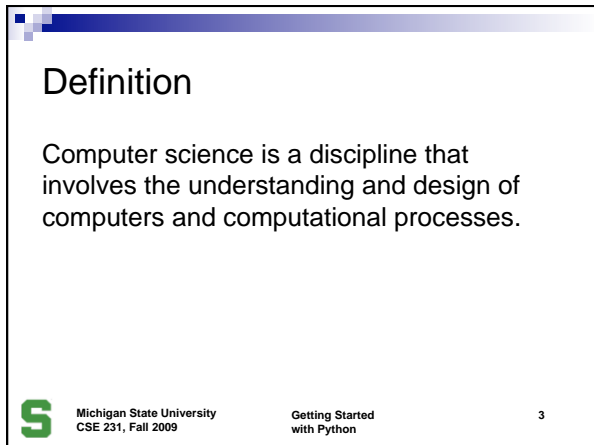


Getting Started

CSE 231, Bill Punch




What is Computer Science?



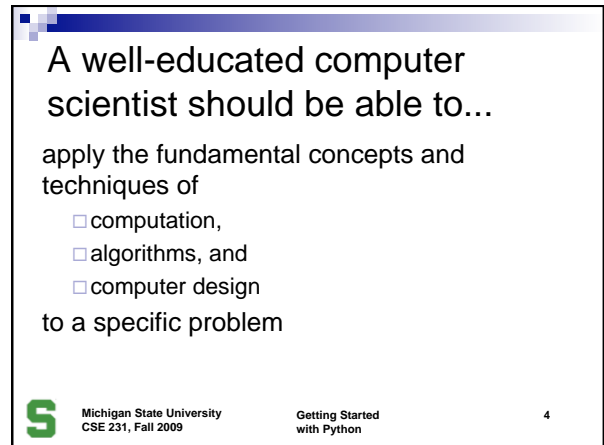
Definition

Computer science is a discipline that involves the understanding and design of computers and computational processes.

 Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

3




A well-educated computer scientist should be able to...

apply the fundamental concepts and techniques of

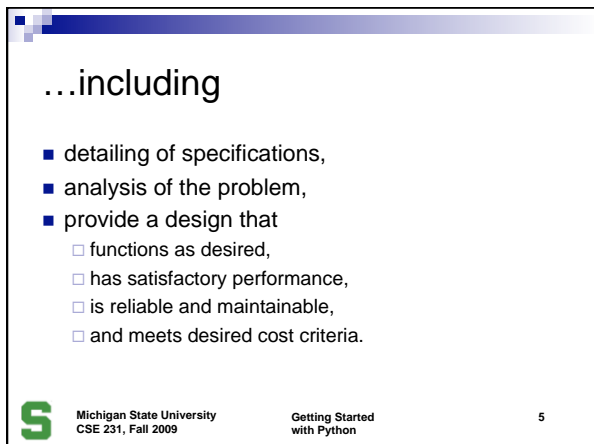
- computation,
- algorithms, and
- computer design

to a specific problem

 Michigan State University
CSE 231, Fall 2009


Getting Started
with Python

4



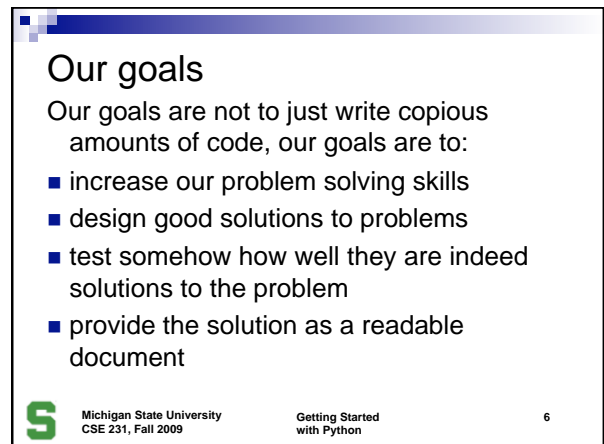
...including

- detailing of specifications,
- analysis of the problem,
- provide a design that
 - functions as desired,
 - has satisfactory performance,
 - is reliable and maintainable,
 - and meets desired cost criteria.

 Michigan State University
CSE 231, Fall 2009

Getting Started
with Python


5



Our goals

Our goals are not to just write copious amounts of code, our goals are to:

- increase our problem solving skills
- design good solutions to problems
- test somehow how well they are indeed solutions to the problem
- provide the solution as a readable document

 Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

6

Getting Python

- Go to <http://www.python.org>
- Go to *Download*
- Download the appropriate version, e.g. Python 2.6.2 Windows installer (or a newer version, if it exists)
- Install it
- Remember, stay away from the 3.1 version!



What is a Computer Program?

Program

- A program is a sequence of instructions.
- To *run* a program is to:
 - create the sequence of instructions according to your design and the language rules
 - turn that program into the binary commands the processor understands
 - give the binary code to the OS, so it can give it to the processor
 - OS tells the processor to run the program
 - when finished (or it dies :-), OS cleans up.



Interpreted

- Python is an *interpreted* language
- interpreted means that Python looks at each instruction, one at a time, and turns that instruction into something that can be run.
- That means that you can simply open the Python interpreter and enter instructions one-at-a-time.
- You can also *import* a program which causes the instructions in the program to be executed, as if you had typed them in.
- To rerun an imported program you *reload* it.



Your First Program Example 1

First Program

```
# first program in python
# input two numbers, add them together, print them out
# wfp, 9/1/07

numStr1 = raw_input('Please enter an integer: ')
numStr2 = raw_input('Please enter a float : ')

int1 = int(numStr1)
float1 = float(numStr2)

print 'The numbers are: ',int1,' and ',float1,' sum is: ',int1+float1
```



Getting input

The function:

```
raw_input("Give me a value")
```

- prints "Give me a value" on the python screen and waits till the user types something (anything), ending with Enter
- Warning, it returns a string (sequence of characters), no matter what is given, even a number (remember, '1' is not the same as 1)



Printing output

```
myVar = 12
```

```
print 'My var has a value of:',myVar
```

- `print` takes a list of elements to print, separated by commas
 - if the element is a string, prints it as is
 - if the element is a variable, prints the value associated with the variable
 - after printing, moves on to a new line of output



At the core of any language

- Control the flow of the program
- Construct and access data elements
- Operate on data elements
- Construct functions
- Construct classes
- Libraries and built-in classes



Save as a "module"

- When you save a file, such as our first program, and place a `.py` suffix on it, it becomes a python module
- You "run" the module from the IDLE menu to see the results of the operation
- A module is just a file of python commands



Errors

- If there are interpreter errors, that is Python cannot run your code because the code is somehow malformed, you get an error
- You can then import the program again until there are no errors



Common Error

- Using IDLE, if you save the file without a `.py` suffix, it will stop colorizing and formatting the file.
- Resave with the `.py`, everything is fine



Syntax

- Lexical components.
- A Python program is:
 - A module (perhaps more than one)
 - Each module has python statements
 - Each statement has expressions



Modules

- We've seen modules already, they are essentially files with Python statements.
- There are modules provided by Python to perform common tasks (math, database, web interaction, etc.)
- The wealth of these modules is one of the great features of Python



Statements

- Statements are commands in Python.
- They perform some action, often called a side effect, but they **do not return any values**



Expressions

- Expressions perform some operation and **return a value**
- Expressions can act as statements, but statements cannot act as expressions (more on this later).
- Expressions typically do not modify values in the interpreter



side effects and returns

- Make sure you get the difference. What is the difference between a side effect and a return?
- $1 + 2$ returns a value (it's an expression). You can "catch" the return value. However, nothing else changed as a result
- print "hello" doesn't return anything, but something else, the side effect, did happen. Something printed!



Python name conventions

- must begin with a letter or `_`
 - `Ab123` is OK, but `123ABC` is not.
- may contain letters, digits, and underscores
 - `this_is_an_identifier_123`
- may be of any length
- upper and lower case letters are different
 - `LengthOfRope` is not `lengthofrope`
- names starting with `_` have special meaning. Be careful



Python comments

- A comment begins with a “#”
- This means that from the “#” to the end of that line, nothing will be interpreted by Python.
- You can write information that will help the reader with the code



Code as essay, an aside

- What is the primary goal of writing code:
 - to get it to do something
 - an essay on my problem solving thoughts
- Code is something to be read. You provide comments to help readability.



Knuth, Literate Programming (84)

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.



Python “types”

- integers: **5**
- floats: **1.2**
- booleans: **True**
- strings: “anything” or ‘something’
- lists: **[,]: ['a', 1, 1.3]**
- others we will see



What is a type

- a type in Python essentially defines two things:
 - the kinds of results you might get
 - the kinds of operations you can perform
- for example, `2/3` is 0, because 2 and 3 are integers
- `'abc'.capitalize()` is a method you can call on strings to capitalize the first letter



Fundamental Types

- Integers
 - `1, -27` (to $+/- 2^{32} - 1$)
 - `123L` L suffix means any length, but potentially very slow. Python will convert if an integer gets too long automatically
- Floating Point (Real)
 - `3.14, 10., .001, 3.14e-10, 0e0`
- Booleans (True or False values)
 - `True, False` note the capital



Example 2

Converting types

- A character '1' is not an integer 1. We'll see more on this later, but take my word for it.
- You need to convert the value returned by the `raw_input` command (characters) into an integer
- `int("123")` yields the integer 123



Type conversion

- `int(someVar)` converts to an integer
- `float(someVar)` converts to a float
- `str(someVar)` converts to a string
- should check out what works:
 - `int(2.1) → 2`, `int('2') → 2`, but `int('2.1')` fails
 - `float(2) → 2.0`, `float('2.0') → 2.0`, `float('2') → 2.0`, `float(2.0) → 2.0`
 - `str(2) → '2'`, `str(2.0) → '2.0'`, `str('a') → 'a'`



import of math

- One thing we did was to import the math module with `import math`
- This brought in python statements to support math (try it in the python window)
- We precede all operations of math with `math.xxx`
- `math.pi`, for example, is pi.
`math.pow(x,y)` raises x to the yth power.



When = doesn't mean equal

- It is most confusing at first to see the following kind of expression:
 - `myInt = myInt + 7`
- You don't have to be a math genius to figure out something is wrong there.
- What's wrong is that = doesn't mean equal



= is assignment

- In many computer languages, = means assignment.
 - `myInt = myInt + 7`
 - `lhs = rhs`
- What "assignment" means is:
 - evaluate all the "stuff" on the rhs of the =
 - take the resulting value and associate it with the name on the lhs



Variable Objects

- Python maintains a list of pairs for every variable:
 - variable's name
 - variable's value
- A variable is created when a value is assigned the first time. It associates a name and a value
- subsequent assignments update the associated value.
- we say name references value
- A variable's type depends on what is assigned.

$X = 7$ →

Name	Value
X	7



Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

37

Namespace

- A namespace is the table that contains the association of a name with a value
- We will see more about namespaces as we get further into Python, but it is an essential part of the language.



Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

38

Assignment Statement

- Example: $x = 2 + 3 * 5$
 - evaluate expression ($2+3*5$): 17
 - change the value of x to reference 17
- Example (y has value 2): $y = y + 3$
 - evaluate expression ($y+3$): 5
 - change the value of y to reference 5



Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

39

More on types

- Python does not require you to pre-define the type of a variable
- What type a variable holds can change
- Nonetheless, knowing the type can be important for using the correct operation on a variable.



Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

40

What can go on the lhs

- There are limits therefore as to what can go on the lhs of an assignment statement.
- The lhs must indicate a name with which a value can be associated
- must follow the naming rules
 - $myInt = 5$, Yes
 - $myInt + 5 = 7$, No



Michigan State University
CSE 231, Fall 2009

Getting Started
with Python

41

Example 3

More on whitespace

- For the most part, you can place “white space” (spaces) anywhere in your program
- use it to make a program more readable

```
1 +  
    2  
- 4
```



continuation

- However, python is sensitive to end of line stuff. To make a line continue, use the \
- ```
print "this is a test", \
" of continuation"
prints
this is a test of continuation
```



## also, tabbing is a special

- The use of tabs is also something that Python is sensitive to.
- We'll see more of that when we get to control, but be aware that the tab character has meaning to Python



## Some of the details

- OK, there are some details you have to get used to.
- Let's look at the syntax stuff
- We'll pick more up as we go along



## Python Tokens

|                                                                   |          |         |        |        |       |
|-------------------------------------------------------------------|----------|---------|--------|--------|-------|
| Keywords:                                                         | and      | del     | from   | not    | while |
| You cannot use (are prevented from using) them in a variable name | as       | elif    | global | or     | with  |
|                                                                   | assert   | else    | if     | pass   | yield |
|                                                                   | break    | except  | import | print  |       |
|                                                                   | class    | exec    | in     | raise  |       |
|                                                                   | continue | finally | is     | return |       |
|                                                                   | def      | for     | lambda | try    |       |



## Python Operators

Reserved operators in Python (expressions)

```
+ - * ** / // %
<< >> & | ^ ~
< > <= >= == != <>
```



## Python Punctuators

- Python punctuation/delimiters (\$ and ? not allowed).

‘ “ # \  
( ) [ ] { } @  
, : . ` = ;  
+= -= \*= /= //= %=  
&= |= ^= >>= <<= \*\*=



## Operators

- Integer
  - addition and subtraction: +, -
  - multiplication: \*
  - division
    - quotient: /
    - remainder: %
- Floating point
  - add, subtract, multiply, divide: +, -, \*, /



## Mixed Types

- You know that  $4/3$  is 1 (integer division)
- You know that  $4.0/3.0$  is 1.3333333
- What is  $4/3.0$ ?
  - no mixed type operations. Must convert
  - Python will automatically convert to the most detailed result. Thus  $4 \rightarrow 4.0$ , the result is 1.3333333



## Collections of data types (later in semester)

- lists
  - sequence of any data elements
- dictionary
  - a list of name:value pairs. Very powerful!
- Class (defines an object when instantiated)
  - a user-defined data type



## Collections and building blocks (examples of what's to come later)

- Point (x,y,z are real numbers)
- Plane (3 points in a plane)
- Line segment (2 end points)
- Face (N points on a plane)
- Solid (N faces)
- Solid (N planes)



## Note

- The interpreter translates Python into machine language. The first stage of that process is determining that it is valid Python code.
- If the program is not a valid Python program, the interpreter cannot translate it.
- If the interpreter cannot translate your code, it is not a Python program so it is not worth any points.



