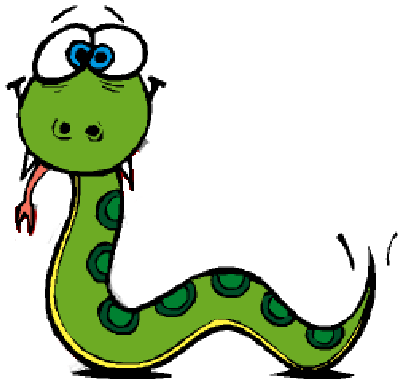
IDCE 302: Chapter 2

Variables, Expressions and Statements

Jie Tian, PhD

Clark University



Outline

- Variables and values
- Naming convention
- Statements, operators, & expressions
- Comments
- Backslash

Variable

- A variable is a name that refers to a value.
- “a box storing a value”
  - Assign variables with values
  - Check the values of variables
  - Check the data type of variables
- A variable does **NOT** need to be declared before used in Python.

# assign variables with values

```
>>> message = "Python version: 2.7"
>>> n = 17
>>> pi = 3.14159
```

# check the values stored in the variables

```
>>> print message
Python version: 2.7
>>> print n
17
>>> print pi
3.14159
```

# check the data types of the variables?

```
>>> type (message)
>>> type (n)
>>> type (pi)
```

Variable Names & Keywords

How to name variables?

- **Meaningful**: document what the variables are used for.
- **Must be legal**: cannot be reserved words.
- Follow the rules of an <u>**identifier**</u>
  - Start with a letter or underscore
  - <u>**Case sensitive!!!**</u>

- Python has **31 keywords:**

```
and        del        from       not        while
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
```

Automatic syntax checking will assist you to find errors.

Do **NOT** use any of the following words either

- They are not strictly Python reserved words
- But they conflict with names of commonly-used Python functions

Data	**Float**	**Int**	**Numeric**	
array	**close**	**float**	**int**	**input**
open	**range**	**type**	**write**	**zeros**

- You should also avoid all the names defined in the **math** library
- You MUST avoid them if you import the library.

acos	**asin**	**atan**	**cos**	**e**
exp	**fabs**	**floor**	**log**	**log10**
pi	**sin**	**sqrt**		

Identifier

- Identifiers are described by the following lexical definitions:

```
identifier ::= (letter|"_") (letter | digit |  "_")

letter    ::=  lowercase | uppercase
lowercase ::=  "a"..."z"
uppercase ::=  "A"..."Z"
digit     ::=  "0"..."9"
```

- Identifier examples (such as variable names, function names in a program):

```
i, j, file_name, _inputX, layer1, value1_x,
value2_x, Calc_area, calc_suitability
```

Operators & Operands

- Operators: special symbols that represent computations like: **+, -, *, /, %**
- Operands: The values an operator uses

Operators	**Name**	**Example**	**Output**
+	Addition	**4+5**	**9**
-	Subtraction	**8-5**	**3**
*****	Multiplication	**4*5**	**20**
/	Division	**19/3**	**6**
%	Remainder	**19%3**	**1**
******	Exponent	**2**4**	**16**

Expressions

- An expression is a combination of **values**, **variables**, and **operators**.

- In interactive mode, the interpreter **evaluates** it and displays the result:

```
>>> 20+32
52
>>> 5**2
25
>>> (1+9)*(8-5)
30
```

Statements

- A **unit of code** (e.g. a line) that the Python interpreter can understand and **execute**.
- A script usually contains a sequence of statements.

```
>>> x = 2
>>> print x
2

>>> message="I am learning Python programming!"
>>> print message
I am learning Python programming!
```

Order of Operations

Parentheses have the highest precedence

2 * (3-1) => 4

(1+1)(5-2) => 8**

Exponentiation has the next highest precedence

21+1 => 3**

3*13 => 3**

Multiplication and ***Division*** have the same precedence (higher than Addition and Subtraction)

2*3-1 => 5

2/3-1 => -1

Note: *Operators with the same precedence are evaluated from left to right*

Operations on Strings

- The **+** operator does work with strings:
- It means “concatenate” here.

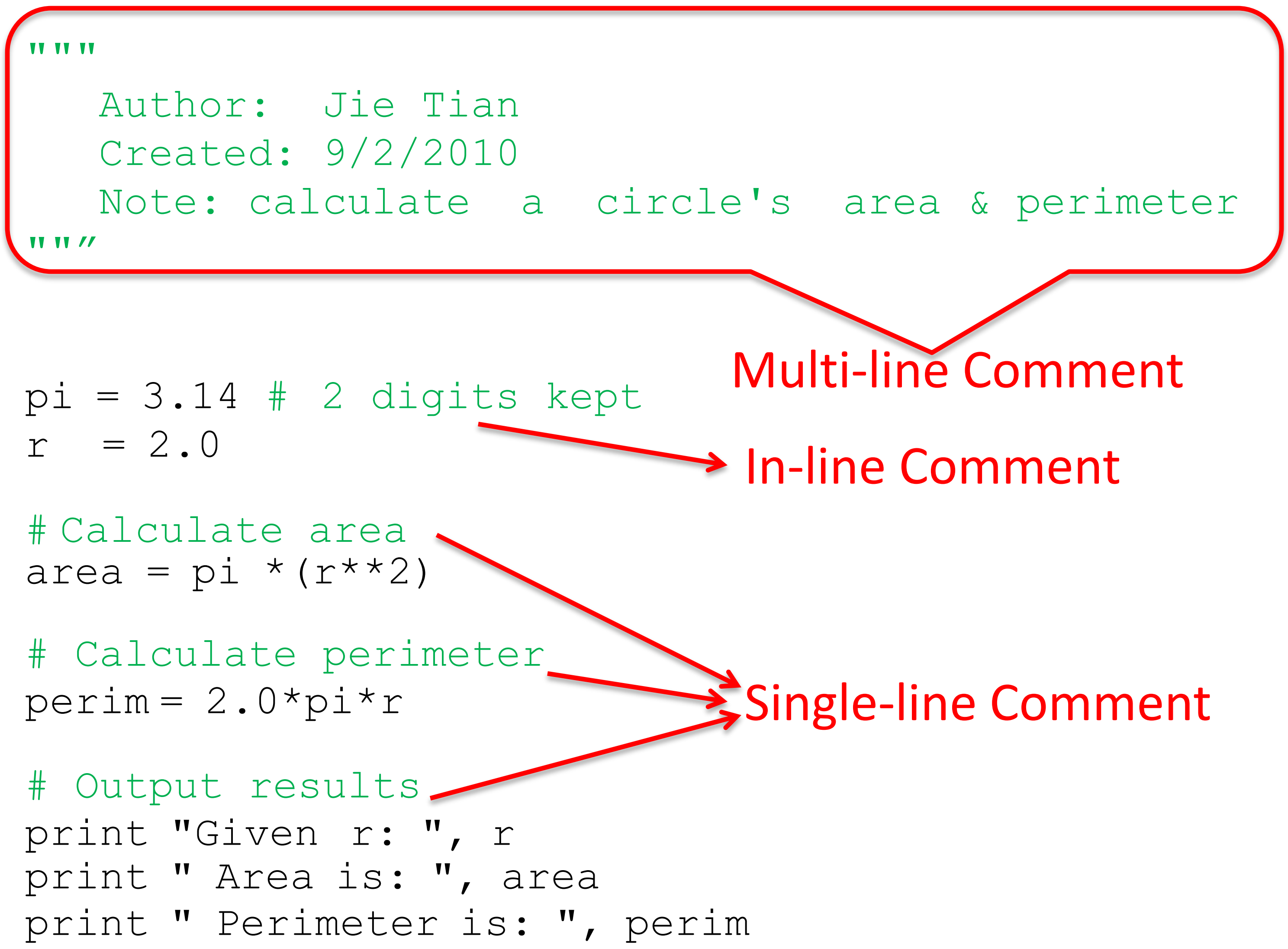
```
>>> folder  = "d:/data/ma/"
>>> name1 = "roads.shp"
>>> name2 = "parks.shp"
>>> road_layer = folder  + name1
>>> park_layer = folder + name2
>>> print road_layer
d:/data/ma/roads.shp
>>> print park_layer
d:/data/ma/parks.shp
```

- The * operator works with strings too.
- But it means “repeat” here.

```
>>>mystring = "point"*3
'poingpointpoint'
```

Comments

- It is often difficult to look at a piece of code and figure out what it is doing.
- It is GOOD to add “notes” (or comments).
- Types of Comments
  - Multi-line comments (a block of notes)
  - Single-line comments
  - In-line comments



```
"""
   Author:   Jie Tian
   Created: 9/2/2010
   Note: calculate  a  circle's  area & perimeter
"""

pi = 3.14 #  2 digits kept
r   = 2.0

# Calculate area
area = pi *(r**2)

#  Calculate perimeter
perim= 2.0*pi*r

#  Output results
print "Given r: ", r
print " Area is: ", area
print " Perimeter is: ", perim
```

Backslash (\), the escape character

- The backslash character (\) is to start character sequences (so named *escape sequences*) which have to be interpreted differently from the same characters occurring alone.
- Examples:

`\n`	`Newline`
`\t`	`Tab`
… …	`(There are a few others)`

- Double Backslashes (\\)

```
>>> filename2 = "d:\data\texas\new_roads.shp"
>>> print filename2
d:\data  exas
ew_roads.shp

>>> filename = "d:\\data\\texas\\new_roads.shp"
>>> print filename
d:\data\texas\new_roads.shp
```

- Output quotation marks
- If I want to output the following line:

  The new film is called "Harry Potter"!

```
>>> str = "The new film is called "Harry Potter"!" ❌

>>> str = "The new film is called \"Harry Potter\"!"
>>> print str
The new film is called "Harry Potter"!
```

Summary

- Variables refer to values (assign, check value or type)
- Follow the naming convention!
- Operators, expressions, & statements
- Write comments
- Understand the use of backslash

Homework document naming:

Examples:

```
MyName_ch2_01.py
MyName_ch2_01.doc

e.g.:
JTian_ch2_01.py
JTian_ch2_01.doc
```

Exercise

Task: create a .py program and run it!

1. Type up the code in a new Python Script window
2. Save it on hard drive and give it a proper name (the file name extension has to be .py)
3. Edit it, add comments
4. Check syntax error
5. Save the file
6. Run the program

```
pi = 3.14
r  = 2.0
area = pi *(r**2)
perim= 2.0*pi*r

print "Given r: ", r
print " Area is: ", area
print " Perimeter is: ", perim
```