

A decorative background pattern on the left side of the slide, consisting of a grid of white geometric shapes (circles and squares) with internal patterns, set against a solid blue background.

Python Coding Schools

2nd lesson

Seed Academy

Agenda

- wk1. Installing Python, HelloWorld
- wk2. Arithmetic Operators
- wk3. Data Types : Integer, Floating point
- wk4. Data Types : String, Boolean
- wk5. Data Structures: List
- wk6. Data Structures: Set, Tuples
- wk7. Data Structures: Dictionary

Agenda

- wk8. Control flows
- wk9. Conditional
- wk10. Loops
- wk11. Function
- wk12. Class
- wk13. Data Visualization

Class materials

<https://github.com/TaeheeJeong/seedacademy>

<https://github.com/TaeheeJeong/SummerCoding2023>

Today's agenda

- Constant
- Variable assignment
- Arithmetic operators
- Quiz

Constants

- Fixed values such as numbers, letters, and strings, are called *constants* because their value does not change.
- Numeric constants are as you expect
- String constants use single quotes (') or double quotes (")

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

Reserved Words

- You cannot use reserved words as variable names / identifiers.

False	class	return	is	finally
None	if	for	lambda	continue
True	def	from	while	nonlocal
and	del	global	not	with
as	elif	try	or	yield
assert	else	import	pass	
break	except	in	raise	

Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable “name”
- Programmers get to choose the names of the variables
- You can change the contents of a variable in a later statement

```
x = 12.2  
y = 14  
x = 100
```


Python Variable Name Rules

- Must start with a letter or underscore _
- Must consist of letters, numbers, and underscores
- Case Sensitive

Good:	spam	eggs	spam23	_speed
Bad:	23spam	#sign	var.12	
Different:	spam	Spam	SPAM	

Sentences or Lines

```
x = 2
```

Assignment statement

```
x = x + 2
```

Assignment with expression

```
print(x)
```

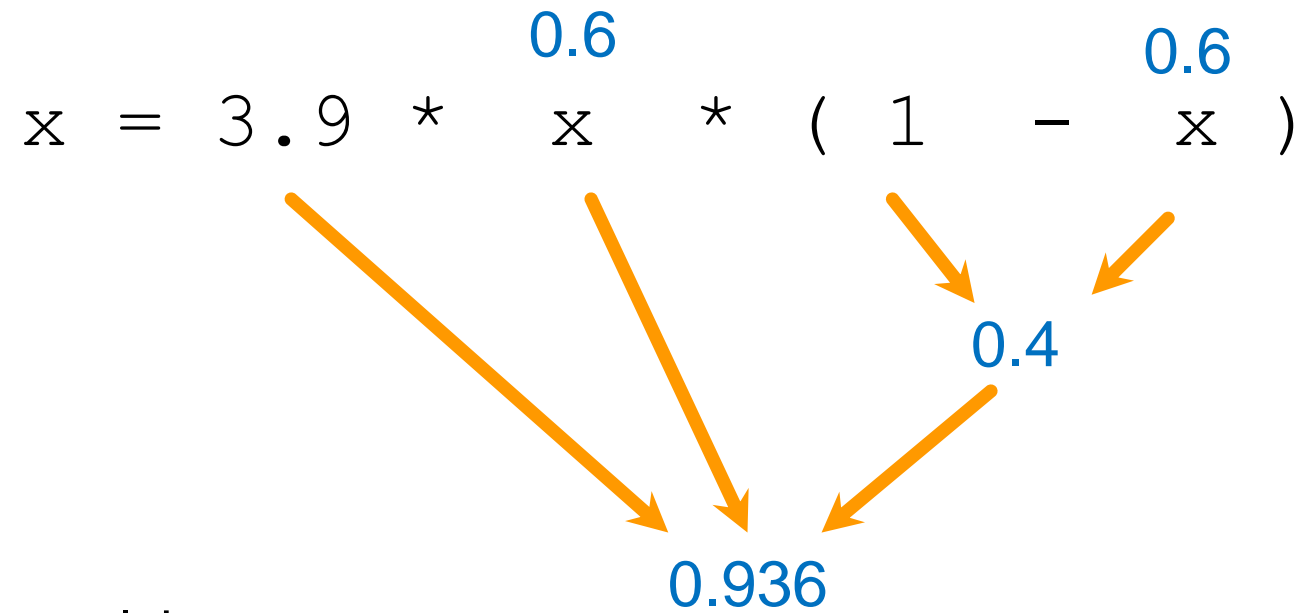
Print statement

Assignment Statements

- We assign a value to a variable using the assignment statement (=)
- An assignment statement consists of an expression on the right-hand side and a variable to store the result

```
x = 3.9 * x * ( 1 - x )
```

A variable is a memory location used to store a value (0.6)



The right side is an expression.
Once the expression is evaluated, the result is placed in (assigned to) x .

A variable is a memory location used to store a value. The value stored in a variable can be updated by replacing the old value (0.6) with a new value (0.936).

$$x = 3.9 * x * (1 - \frac{0.6}{x})$$

The diagram illustrates the evaluation of the expression $x = 3.9 * x * (1 - \frac{0.6}{x})$. It shows the intermediate steps: the values 3.9, x (0.6), 1, and 0.6 are combined to form the intermediate result 0.4, which is then multiplied by 3.9 to yield the final result 0.936.

The right side is an expression. Once the expression is evaluated, the result is placed in (assigned to) the variable on the left side (i.e., x).

Arithmetic Operators

- Because of the lack of mathematical symbols on computer keyboards - we use **computer-speak** to express the classic math operations
- Asterisk is multiplication
- Exponentiation (raise to a power) looks different than in math

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Remainder

Arithmetic Operators

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4

>>> yy = 440 * 12
>>> print(yy)
5280

>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3

>>> print(4 ** 3)
64
```

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Remainder

Order of operators

- When we string operators together - Python must know which one to do first
- This is called “operator precedence”
- Which operator “takes precedence” over the others?

$x = 1 + 2 * 3 - 4 / 5 ** 6$

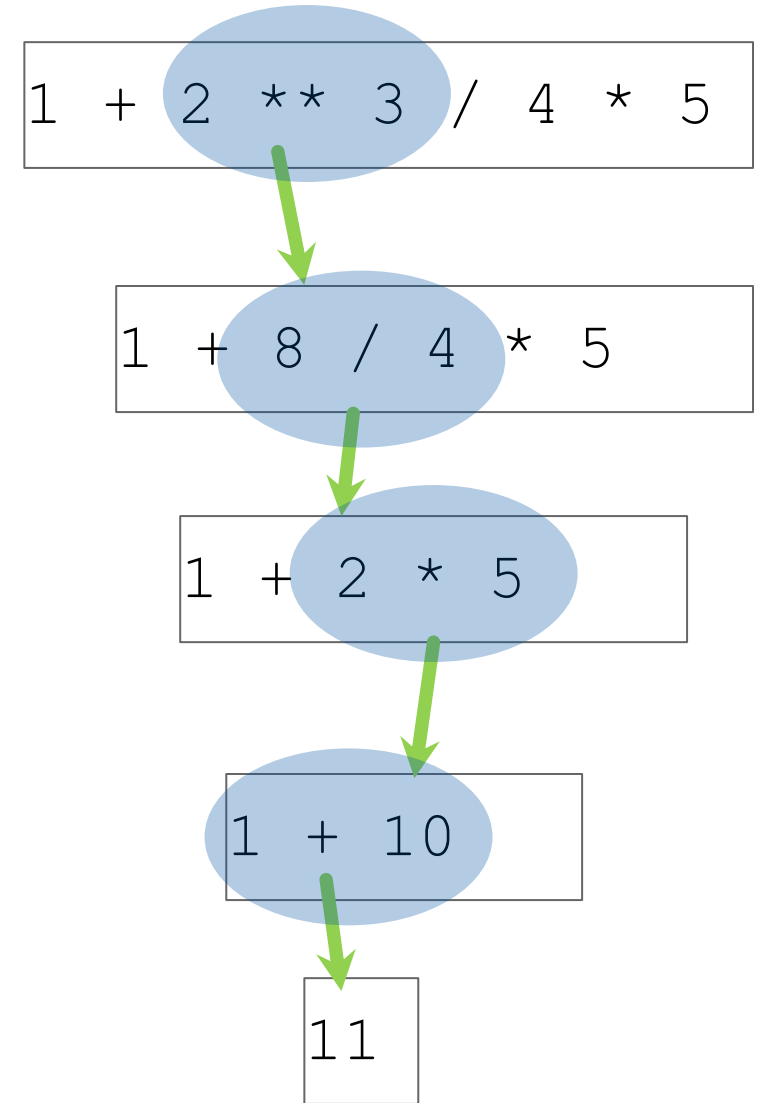

Operator Precedence Rules

Highest precedence rule to lowest precedence rule:

- Parentheses are always respected
- Exponentiation (raise to a power)
- Multiplication, Division, and Remainder
- Addition and Subtraction
- Left to right

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
```

Parenthesis
Power
Multiplication
Addition
Left to Right



Comments in Python

- Anything after `#` is ignored by Python
- Why comment?
 - Describe what is going to happen in a sequence of code
 - Document who wrote the code or other ancillary information
 - Turn off a line of code - perhaps temporarily

Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Modification: Taehee Jeong, San Jose State University