Python

CS*101* `lec01`

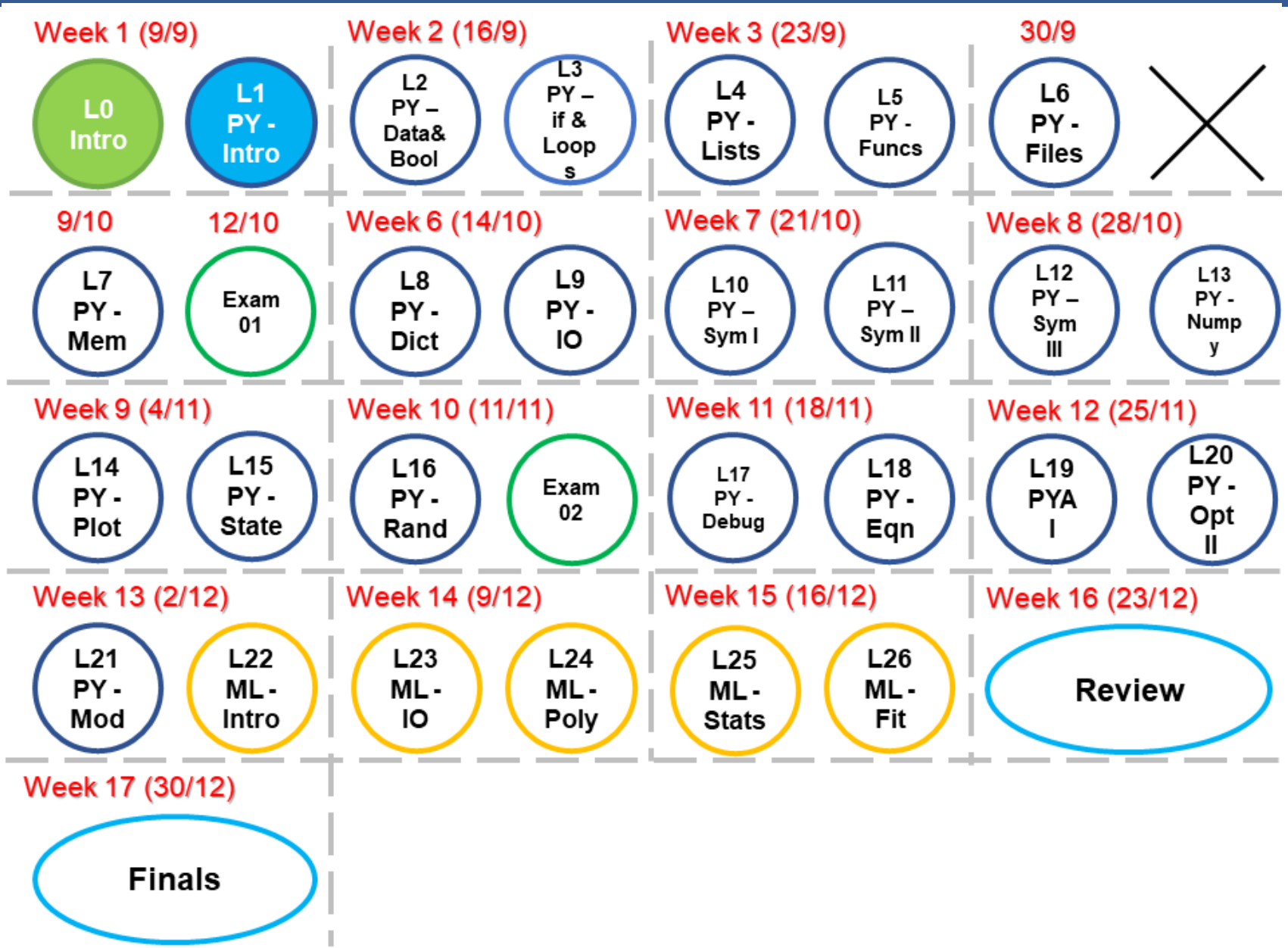
Introduction to Python

2019-09-11

Announcements

- Have you signed up in RELATE yet?
  https://relate.intl.zju.edu.cn/course/2019Fall-CS101/
- `quiz00` *due today and* `quiz01` *due tomrrow*

Roadmap


Week 1 (9/9)
L0 Intro
L1 PY - Intro
Week 2 (16/9)
L2 PY – Data& Bool
L3 PY – if & Loop s
Week 3 (23/9)
L4 PY - Lists
L5 PY - Funcs
30/9
L6 PY - Files
9/10
L7 PY - Mem
12/10
Exam 01
Week 6 (14/10)
L8 PY - Dict
L9 PY - IO
Week 7 (21/10)
L10 PY – Sym I
L11 PY – Sym II
Week 8 (28/10)
L12 PY – Sym III
L13 PY - Numpy
Week 9 (4/11)
L14 PY - Plot
L15 PY - State
Week 10 (11/11)
L16 PY - Rand
Exam 02
Week 11 (18/11)
L17 PY - Debug
L18 PY - Eqn
Week 12 (25/11)
L19 PYA I
L20 PY - Opt II
Week 13 (2/12)
L21 PY - Mod
L22 ML - Intro
Week 14 (9/12)
L23 ML - IO
L24 ML - Poly
Week 15 (16/12)
L25 ML - Stats
L26 ML - Fit
Week 16 (23/12)
Review
Week 17 (30/12)
Finals

Review

Question

Nothing to review! Yeh!

Objectives

1. Compose expressions and statements using Python syntax.
2. Identify and correct simple errors in mathematical expressions.
3. Annotate short programs in Python with comments. container.
4. Use Python or IPython (e.g., Jupyter)

Program

What is a program?

What is a program?

- A set of instructions a computer executes to achieve a goal.

What is a program?

- A set of instructions a computer executes to achieve a goal.
- For us, “programming” = “computing” = “coding”.

What is data?

What is data?

- Information stored in a computer.

What is data?

- Information stored in a computer.
- Bit and Byte

What is data?

- Information stored in a computer.
- Bit and Byte
  - 1 bit = 0 or 1

What is data?

- Information stored in a computer.
- Bit and Byte
  - 1 bit = 0 or 1
  - 1 byte = 8 bits = XXXXXXXX where X can be 0 or 1

What is data?

- Information stored in a computer.
- Bit and Byte
  - 1 bit = 0 or 1
  - 1 byte = 8 bits = XXXXXXXX where X can be 0 or 1
  - 00000010 (or simply 10) = 2

What is data?

- Information stored in a computer.
- Bit and Byte
  - 1 bit = 0 or 1
  - 1 byte = 8 bits = XXXXXXXX where X can be 0 or 1
  - 00000010 (or simply 10) = 2
  - 00011101 (or simply 11101) = ?

What is data?

- Information stored in a computer.
- Bit and Byte
  - 1 bit = 0 or 1
  - 1 byte = 8 bits = XXXXXXXX where X can be 0 or 1
  - 00000010 (or simply 10) = 2
  - 00011101 (or simply 11101) = ?
    - 29

$$29_{10} = 11101_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

1×2^4 1×2^3 1×2^2 0×2^1 1×2^0

What is data?

- Information stored in a computer.
- Bit, Byte, Word

What is data?

- Information stored in a computer.
- Bit, Byte, Word
- All data is stored in binary.

What is data?

- Information stored in a computer.
- All data is stored in binary.

```
-------- --- --- -------- ---------
00000001 00101010 01000000 00100000
```

What is data?

- Binary data must be interpreted:
  - value (number, character)

What is data?

- Binary data must be interpreted:
  - value (number, character)
  - memory location
  - instruction

What is data?

- Binary data must be interpreted:
  - value (number, character)
  - memory location
  - instruction

What is a program?

- So to a computer, a program is just data!

What is a program?

- So to a computer, a program is just data!
- Instructions are encoded in binary.

```
-------- --------- -------- --------
00000001 00101010 01000000 00100000
         add $t0, $t1, $t2
```

What is a program?

- Programs are data!
- Instructions are encoded in binary.

```
-------- --------- -------- --------
00000001 00101010 01000000 00100000
         add $t0, $t1, $t2
              =B5+B6
```

What is a program?

- Programs are data!
- Instructions are encoded in binary.

```
-------- --------- -------- --------
00000001 00101010 01000000 00100000
         add $t0, $t1, $t2
              =B5+B6
          x = y + z
```

What is a program?

- Programs are data!
- Instructions are encoded in binary.

```
-------- --------- -------- --------
00000001 00101010 01000000 00100000
add $t0, $t1, $t2
=B5+B6
x = y + z
```

- High-level languages express things more like humans.
- Low-level languages are “closer to the metal (CTM)”.

Elements of Programs

A simple program

DEG F	11
DEG C	(DEG F -32)*5/9
TEST: 100 C =	212 F

Syntax

```
F = 212                 # deg F
C = (F-32) * 5/9        # deg C
print( F,' deg F is ',C,' deg C.' )
assert C == 100
```

212 deg F is 100 deg c

A simple program

- 0. Headers for compilers

A simple program

- 0. Headers for compilers
- 1. Declare variables.
  - E.g., int a; float b; (not needed in Py n Matlab)
- 2. Initialize variables.
  - E.g., a = 10

A simple program

- 0. Headers for compilers
- 1. Declare variables.
  - E.g., int a; float b; (not needed in Py n Matlab)
- 2. Initialize variables.
  - E.g., a = 10
- 3. Alogrithm. Including

A simple program

- 0. Headers for compilers
- 1. Declare variables.
  - E.g., int a; float b; (not needed in Py n Matlab)
- 2. Initialize variables.
  - E.g., a = 10
- 3. Alogrithm. Including
  - Built-in functions/operators
  - Define own functions
  - Loops
  - Logics
  - Input
  - ...

A simple program

- 0. Headers for compilers
- 1. Declare variables.
  - E.g., int a; float b; (not needed in Py n Matlab)
- 2. Initialize variables.
  - E.g., a = 10
- 3. Alogrithm. Including
  - Built-in functions/operators
  - Define own functions
  - Loops
  - Logics
  - Input
  - ...
- 4. Output (to screen/file or another function)

A simple program

- 0. Headers for compilers
- 1. Declare variables.
  - E.g., int a; float b; (not needed in Py n Matlab)
- 2. Initialize variables.
  - E.g., a = 10
- 3. Alogrithm. Including
  - Built-in functions/operators
  - Define own functions
  - Loops
  - Logics
  - Input
  - ...
- 4. Output (to screen/file or another function)

A simple program

Initialize variables

Function block: Using operators

```
F = 212                # deg F
C = (F-32) * 5/9       # deg C
print( F,' deg F is ',C,' deg C.' )
assert C == 100
```

Output

A simple program

```
F = 212                # deg F
C = (F-32) * 5/9       # deg C
print( F,' deg F is ',C,' deg C.' )
assert C == 100
```

1. Note the structure here.
2. Structure—syntax—is fixed!
3. Syntax is how to write a program, like spelling and grammar.

A simple program

```
Writing a simple Python program is like
writing an English essay. There are very
precise rules for grammar and spelling. The
hardest part is learning how to translate
one's thoughts into this precise mode of
expression.
```

Syntax

```
F = 212                # deg F
C = (F-32) * 5/9       # deg C
print( F,' deg F is ',C,' deg C.' )
assert C == 100
```

Several elements of Python syntax in this program:

Syntax

```
F = 212                 # deg F
C = (F-32) * 5/9        # deg C
print( F,' deg F is ',C,' deg C.' )
assert C == 100
```

Several elements of Python syntax in this program:

1. `literals` - Numbers and the text strings (blue and red)
2. `keywords` - Commands or structural indicators
3. `names` - variables or functions like *print* and *C*
4. `operators` - like + and = calculate values
5. `comments` - # anything here is not interpreted by the machine

What is a literal?

- Fixed value (noun)
- Represents data that doesn't change (`3` or `'firefly'`)

What is an operator?

- Manipulates data (verb)
- + - * **

What is an expression?

- Combines literals and operators (phrase)

What is an expression?

- Combines literals and operators (phrase)
- Produce a new value
  - `3 * 5`
  - `100 - 23`

What is an expression?

- Can be arbitrarily complicated
  - `3 + 8*5 + 4 - 7/100`

Question

$\texttt{1} + \texttt{1} * \texttt{2} \stackrel{?}{=}$

a) 4

b) 3

c) Something else

Question

$\texttt{1} + \texttt{1} * \texttt{2} \stackrel{?}{=}$

a) 4

b) 3 $\star\star$`Order of operations`

c) Something else

Question

$23 + 6/2 - 4 \stackrel{?}{=}$

a) 22

b) 18

c) -9

d) None of these are correct.

Question

$\texttt{23} + \texttt{6}/\texttt{2} - \texttt{4} \stackrel{?}{=}$

a) 22 ⋆⋆`Again, ooooo....`

b) 18

c) -9

d) None of these are correct.

Use parentheses!

$23 + (6/2) - 4$ is always clearer.

What are some other operators?

- exponentiation, `**`

What are some other operators?

- exponentiation, `**`
- modulus, `%` (gives the remainder)

What are some other operators?

- exponentiation, `**`
- modulus, `%` (gives the remainder)
- floor division, `//`

More operators? (Optional)

- bitwise `OR`, `|`
- bitwise `XOR`, `^`
- bitwise `AND`, `&`
- bitwise left shift, `<<`
- bitwise right shift, `>>`

More operators? (Optional)

- bitwise `OR`, `|`
- bitwise `XOR`, `^`
- bitwise `AND`, `&`
- bitwise left shift, `<<`
- bitwise right shift, `>>`
- You don't need to know these, but—

Example (Optional)

`1 ^ 2` $\stackrel{?}{=}$

a) 0

b) 1

c) 2

d) 3

Example (Optional)

`1 ^ 2` $\stackrel{?}{=}$

a) 0

b) 1

c) 2

d) 3 $\star$! `XOR`, NOT exponentiation!

Expression... so what?

$23 + (6/2) - 4$

- An expression does not change the machine state.

Expression... so what?

$23 + (6/2) - 4$

- An expression does not change the machine state.
- Programs are complex, and we need to remember results for later use.

How do we reuse values?

- Low-level languages refer directly to memory address:

```
ADD DATA AT        10101101 11010100
TO DATA AT         11010100 01001001
STORE RESULT AT    00001101 01001110
```

What is a variable?

- The solution: give a name to memory locations!

What is a variable?

- The solution: give a name to memory locations!
- Variables name a memory location

What is a variable?

- The solution: give a name to memory locations!
- Variables name a memory location
- Variables store a value

What is a variable?

- The solution: give a name to memory locations!
- Variables name a memory location
- Variables store a value
- This value can change over time — it is a placeholder.

What new operator do we need to

- To store a value, we need...

What new operator do we need to

- To store a value, we need...
- assignment, = (single equals sign)

Example

What value is stored in the variable `x`?

```
x = 17 + 7*9
```

a) 3
b) 31
c) 80
d) 216

Example

What value is stored in the variable `x`?

```
x = 17 + 7*9
```

a) 3
b) 31
c) 80 ⋆
d) 216

Example

What value is stored in the variable `x`?

```
x = 17 + 7*9
x = 3
```

a) 0
b) 1
c) 2
d) 3

Example

What value is stored in the variable `x`?

```
x = 17 + 7*9
x = 3
```

a) 0
b) 1
c) 2
d) 3 ⋆

What is a statement?

- A statement changes the state of the computer (sentence)

What is a statement?

- A statement changes the state of the computer (sentence)
- Example: an assignment
- `x = 17 + 7*9`

A first program

```
x = 10
y = x ** 2
y = y + y
```

What is a comment?

- A comment is ignored by the interpreter

What is a comment?

- A comment is ignored by the interpreter
- Example: `# anything after a hash`
- So why do we need a comment ???

What is a keyword?

- A keyword is a reserved word with a special meaning to Python
- Shown with bold or coloured type.

What is a keyword?

- A keyword is a reserved word with a special meaning to Python
- Shown with bold or coloured type.
- Example: `for`, `in`, `assert`

How can I use Python?

- Obtain a distribution of Python 3.
  - We recommend Anaconda. See course website

How can I use Python?

- Obtain a distribution of Python 3.
  - We recommend Anaconda. See course website
  - Py2 vs Py3. `print` needs parentheses!
  - Use Py3 !!!

How can I use Python?

- Write code in one of three ways:

How can I use Python?

- Write code in one of three ways:
  - Directly (`python.exe`)

How can I use Python?

- Write code in one of three ways:
  - Directly (`python.exe`)
  - Script (text editor or Spyder)

How can I use Python?

- Write code in one of three ways:
  - Directly (`python.exe`)
  - Script (text editor or Spyder)
  - Jupyter (IPython) notebook (as in labs)

Summary

- Programs consist of series of `statements`.

Summary

- Programs consist of series of `statements`.
- Each statement is executed normally in order from top to bottom.

Summary

- Programs consist of series of `statements`.
- Each statement is executed normally in order from top to bottom.
- A `statement` contains `Literal, operators, expressions`, and `keywords`.

Summary

- Programs consist of series of `statements`.
- Each statement is executed normally in order from top to bottom.
- A `statement` contains `Literal, operators, expressions`, and `keywords`.
- Remember to include your `comments`