Introduction to Python

Instructor: Anika

Introductions

Where are you from?

Why did you opt for the course?

Interesting thing about you?

Your field of studies/profession?

Programming Background?

What is Python?

• Code /Source code: The sequence of instructions in a program. •

•Syntax: The set of legal structures and commands that can be used in a particular programming language.

•Output: The messages printed to the user by a program.

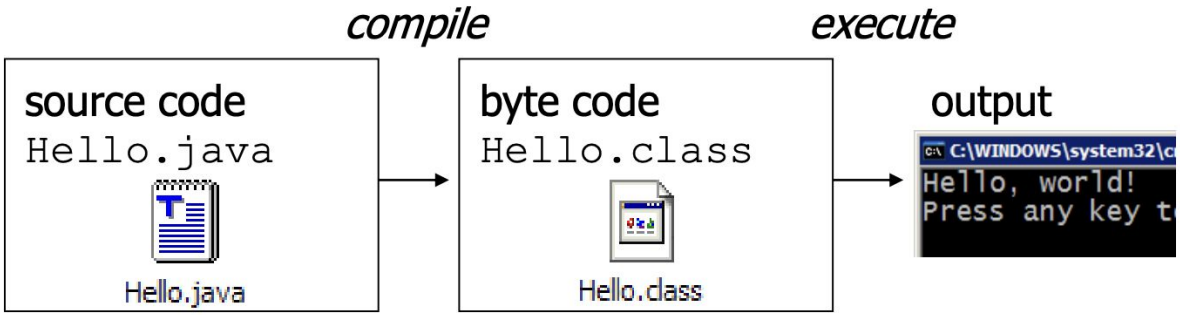
• Console: The text box onto which output is printed. Some source code editors pop up the console as an external window, and others ~~cont~~ain their own console window.

Basic Data Types

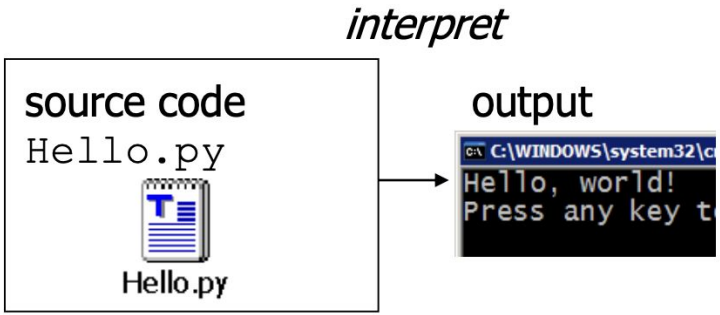
Integer	Boolean	Text
Float	True/False	String/str
Int	1/0	

Compiling vs Intewrpreting?

- Many languages require you to *compile* (translate) your program into a form that the machine understands.



- Python is instead directly *interpreted* into machine instructions.



- Python is an interpreted language
- The interpreter provides an interactive environment to play with the language
- Results of expressions are printed on the screen

```
>>> 3 + 7
10
>>> 3 < 15
True
>>> 'print me'
'print me'
>>> print 'print me'
print me
>>>
```

- **expression**: A data value or set of operations to compute a value.

  Examples: `1 + 4 * 3`

  `42`

- Arithmetic operators we will use:

`+ - * /`	addition, subtraction/negation, multiplication, division
`%`	modulus, a.k.a. remainder
`**`	exponentiation

- **precedence**: Order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`

    `1 + 3 * 4` is `13`

  - Parentheses can be used to force a certain order of evaluation.

    `(1 + 3) * 4` is `16`

- When we divide integers with `/` , the quotient is also an integer.

```
    3
4 ) 14
    12
    --
     2
```

```
     52
27 ) 1425
     135
     ---
      75
      54
      --
      21
```

  - More examples:
    - `35 / 5` is `7`
    - `84 / 10` is `8`
    - `156 / 100` is `1`

- The `%` operator computes the remainder from a division of integers.

```
      3
  ----
4 ) 14
    12
    --
     2
```

```
      43
  -----
5 ) 218
    20
    --
     18
     15
     --
      3
```

- Python can also manipulate real numbers.
  - Examples: `6.022` `-15.9997` `42.0` `2.143e17`

- The operators `+ - * / % ** ( )` all work for real numbers.
  - The `/` produces an exact answer: `15.0 / 2.0` is **`7.5`**
  - The same rules of precedence also apply to real numbers: Evaluate `( )` before `* / %` before `+ -`

- When integers and reals are mixed, the result is a real number.
  - Example: `1 / 2.0` is `0.5`
  - The conversion occurs on a per-operator basis.

```
7 / 3 * 1.2 + 3 / 2
  2   * 1.2 + 3 / 2
    2.4     + 3 / 2
    2.4     +   1
        3.4
```

- Python has useful commands (or called functions) for performing calculations.

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`, `***value2***`)`	larger of two values
`min(`***value1***`, `***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

Type Casting

Thank you

Questions