UNIT 2A
An Introduction to Programming

Python

- Python is one of many programming languages.
- 2 widely used versions. We will use Python 3. (Specifically, Python version 3.3.2)
- Running Python on the command line:

  > python3 -i filename.py

  (-i means interactive mode)
Arithmetic Expressions

- **Mathematical Operators**
  - Addition (+)
  - Subtraction (-)
  - Multiplication (*)
  - Division (/)
  - Modulo (%)
  - Exponentiation (**)

- **Order of Precedence**
  - **Exponentiation**: \(a^b\)
  - **Multiplication and Division**: \(a \times b\)
  - **Addition and Subtraction**: \(a \pm b\)

- Use parentheses to force alternate precedence:
  - \(5 \times 6 + 7 \neq 5 \times (6 + 7)\)
  - \(2 + 3 + 4 = (2 + 3) + 4\)
  - \(2 \times 3 \times 4 = 2 \times (3 \times 4)\)

Data Types

- **Integers**
  - 4 15110 -53 0
- **Floating Point Numbers**
  - 4.0 -0.8 0.3333333333333333 7.34e+014
- **Strings**
  - "hello" "A" " " " " "7up!"
  - 'there' ' ' '15110'
- **Booleans**
  - True False

George Boole, 1815-1864
Integer Division

In Python3:
• $7 / 2$ equals $3.5$
• $7 // 2$ equals $3$
• $7 // 2.0$ equals $3.0$
• $7.0 // 2$ equals $3.0$
• $-7 // 2$ equals $-4$ (beware! $//$ rounds down)

Modulo

In Python3:
• $7 \% 2$ equals $1$
• $15 \% 4$ equals $3$
• $42 \% 7$ equals $0$
• $6 \% 14$ equals $6$
• $-7 \% 2$ equals $1$ (think about it...)
Variables

- All variable names must start with a letter (lowercase recommended).
- The remainder of the variable name (if any) can consist of any combination of uppercase letters, lowercase letters, digits and underscores (_).
- Variables are case sensitive. Example: Value is not the same as value.

Using predefined modules

- `math` is a predefined module of methods (functions) that we can use without writing the implementations.

```python
import math
math.sqrt(16)
math.pi
math.sin(math.pi / 2)
```

- We must `import math` before we can use the math functions.
Assignment Statements

• The lefthand side must contain a single variable.
• The righthand side can be any valid Python expression:
  • A numerical, string or boolean value.
    \[ x = 45.2 \]
  • A numerical expression.
    \[ y = x \times 15 \]
  • A method (function) call.
    \[ z = \text{math.sqrt}(15110) \]
  • Any combination of these:
    \[ \text{root1} = -b + \text{math.sqrt}(b*b-4*a*c)/(2*a) \]

Methods

• Methods are used to capture small algorithms that might be repeated with different initial conditions.
  ```python
  def methodname (parameterlist) :
    instruction1
    instruction2
    etc.
  ```
• `def` is a reserved word and cannot be used as a variable name.
• *Indentation is critical.* Use spaces only.
Methods (cont’d)

• The name of a method follows the same rules as names for variables.
• The parameter list can contain 1 or more variables that represent data to be used in the method’s computation.
  • A method can have 0 parameters.
    
    ```python
    def hello_world():
        print("Hello World!\n")
    end
    ```

    (\n is a newline character)

```python
def tip(total):
    return total * 0.18
```

To run the function `tip` in `python3`:
```
python3 -i tip.py
```

```python
>>> tip(100)
⇒18.0

>>> tip(135.72)
⇒24.4296
```
Example: Countertop

Determine the area of a countertop that is a square with a triangle cut out of one of its corners.

```
def compute_area(side):
    square = side * side
    triangle = 0.5 * side / 2 * side / 2
    area = square - triangle
    return area
```

To run the function in python3:
```
python3 -i countertop.py
```
```
>>> compute_area(109)
```

(run function with side = 109)
(note: there are no units)
Methods (cont’d)

• To run a method, we say we “call” the method.
• A method can return either one answer or no answer to its “caller”.
• The hello_world function does not return anything to its caller. It simply prints something on the screen.
• The compute_area function does return its result to its caller so it can use the value in another computation:
  \[\text{compute\_area}(109) + \text{compute\_area}(78)\]

Methods (cont’d)

• Suppose we write compute_area2 this way:
  
  ```python
  def compute_area2(side):
      square = side * side
      triangle = 0.5 * side/2 * side/2
      area = square - triangle
      print area
  ```

  • Now this computation does not work since each function call prints but returns nothing:
    \[\text{compute\_area2}(109) + \text{compute\_area2}(78)\]
Caution: return vs. print

• When you return a result from a function, the caller of that function can use that result in another computation.
  >>> x = 15 + compute_area(110)

• When you print a result in a function, the user will see the result on the screen, but the caller of that function won’t get anything back so it cannot use the result in another computation.
  >>> x = 15 + compute_area2(110)

escape.py
(a function with two parameters)

```python
import math
def compute_ev(mass, radius):
    # computes escape velocity
    univ_grav = 6.67e-011
    return math.sqrt(2*univ_grav*mass/radius)
```

Comments begin with #

To run the function for Earth in python3:

```bash
python3 -i escape.py
>>> compute_ev(5.9742e+024, 6378.1)
35348592957.826279
```
Cautions

- Python has no idea what units you’re using for computations, so data must be given in the proper units or the results are meaningless.
- When you call a function, the number of arguments you supply must match the number of parameters the function requires.
- When you call a function, if you reverse the arguments, Python won’t catch this error: `compute_ev(6378.1, 5.9742e+024)`

Printing multiple things on one line

```python
python3 -i tip.py
>>> print("My tip is ", tip(19.95))
```

In `tip.py`:

```python
def tip(total):
    print("$", tip(19.95), "is my tip.")
    return None
```
import math

def compute_ev(mass, radius):
    # computes escape velocity
    univ_grav = 6.67e-011
    return math.sqrt(2*univ_grav*mass/radius)

def main():
    print compute_ev(6378.1, 5.9742e+024)
main()

> python escape2.py
35348592957.8
>
On the command line, run this command (no –i flag)

Store this program in the file escape2.py

Stand-alone programs (non-interactive)