Fundamental Python Concepts

CSC148, Introduction to Computer Science
Fall 2016
Diane Horton and David Liu
(taught mostly via live programming)
For each example, does $y$ change?

# Example 1:
```python
>>> x = 3
>>> y = 3
>>> x = 5
```

# Example 2:
```python
>>> x = [1, 2, 3]
>>> y = [1, 2, 3]
>>> x[1] = 100
```

# Example 3:
```python
>>> x = 3
>>> y = x
>>> x = 5
```

# Example 4:
```python
>>> x = [1, 2, 3]
>>> y = x
>>> x[1] = 100
```
CSC108 teaches a “recipe” for writing functions (and methods):

1. Write examples of calls and the expected returned values.
2. Write a type contract that identifies the return value and the type of each parameter.
3. Write the function header.
4. Add a one-line summary of what the function does.
5. Write the body.
The type contract

• We will use this style of type annotation:
  @type parameter: type
  @rtype: type
  (“return type”)

• Different from what some saw in csc108.

• Allows pycharm to check that your code conforms.
Docstrings

• Steps 1, 2 and 4 yield a docstring. Example:

```python
def decreases_at(L):
    """Return the index of the first int in L that is less than its predecessor."

    @type L: list[int]
    @rtype: int

    >>> decreases_at([3, 6, 9, 12, 2, 1, 8, 5])
    4
    """
```
Good docstrings

• Principles:
  – Describe **what** the method does, very precisely.
  – Mention every parameter by name.
  – Do not discuss **how** the method works
    E.g., Do not discuss local variables, helper methods, or algorithms.

• Two uses for a docstring:
  – Guides you in writing the body.
  – Defines the interface so callers of the function know how to use it.
Bad docstrings

• A docstring is a specification for the function.
• If the specification is not complete
  – the caller’s expectations are not met and
  – bugs are likely.
• It helps to think of the docstring as specifying a contract.
Good docstrings include preconditions

- If arguments to the function must satisfy requirements (other than the type contract), say so.
- We call these requirements as **preconditions**.
- Label them as such in the docstring.
Design by Contract

CSC148, Introduction to Computer Science
Fall 2016
Diane Horton and David Liu

UNIVERSITY OF TORONTO
Contracts in real life

• A binding agreement between two parties.

• Example:
  Provided that the client arrives 2 hours prior to departure, having paid the full fair in advance, and with luggage that is within set limits, the airline will fly the client and luggage to their destination.

• What if
  – Arrives 3 hours prior to departure?
  – Arrives 10 minutes prior to departure?
  – Brings luggage that exceeds the limits?
What the contract means

• If the client meets the conditions:
  The promised actions will occur.

• If not:
  No guarantees!
Contracts in programming

• A **precondition** for a function is a boolean expression that describes restrictions on the values of the arguments.

• If the arguments satisfy the type contract and the preconditions:
  The function will
  1. halt without crashing, and
  2. do what the docstring says.

• If not:
  No guarantees!
Design by contract

• **Design by contract** is a way of thinking about programming.

• Each function (or method) provides a service.

• Its service is specified as a contract by the docstring, including:
  – What it expects (the precondition)
  – What it guarantees (what the function does)
A little history

• Bertrand Meyer invented the metaphor in 1986.
• Part of his work on the Eiffel programming language.
• An `assert` statement lets you make the precondition executable.
• Eiffel, and many languages since, has assertions.
• You saw them in Python when you used `unittest`.
Uses for a function’s contract

• Guides you in writing the body.
• Defines an interface for callers of the function.
• Aids in debugging.
• Aids in code maintenance.
  – If you improve the implementation, you know what you must ensure is still guaranteed.
• Helps you argue that the code is correct.
  – Defines the standard against which to measure.
  – Is the basis for a proof of correctness.