Python Memory Model

CSC148, Introduction to Computer Science
Fall 2017
Diane Horton
Recap of your reading

• An object has: unique ID, type, value.
• Objects of a mutable type can change value.
• Objects of an immutable type cannot.
• A variable always stores a reference to an object.
• A variable has no type.
• Two variables can reference the same object.
• Understanding the implications allows us to predict what code will do.
print(a)
b.change(...)  # No mention of a
print(a)
How function calls are tracked

• Each call creates a *frame* that stores needed info.
• As one function calls another, these pile up in a stack.
• This is the *call stack*.
Function call

1. A new frame is placed on top of the call stack.
2. Each parameter is defined in that frame.
3. Each argument is evaluated and its value assigned to the corresponding parameter.
```python
def mess_about(n, s):
    message = s * n
    print(message)

If __name__ == '__main__':
    count = 3
    word = 'no'
    mess_about(count, word)
    print('All done.')
```
Function execution

• Then the function body is executed.
• The frame stores any new variables.
• These are local variables.
def mess_about(n, s):
    message = s * n
    print(message)

if __name__ == '__main__':
    count = 3
    word = 'no'
    mess_about(count, word)
    print('All done.')
Returning from a function

• The function ends when either
  – a return is reached, or
  – the end of the function is reached.

• The stack frame for this call is deleted.

• If there was a return, its expression is evaluated and the result returned to the caller.

• Control returns to where the function was called.
def mess_about(n, s):
    message = s * n
    print(message)

if __name__ == '__main__':
    count = 3
    word = 'no'
    mess_about(count, word)
    print('All done.')
Function Design

CSC148, Introduction to Computer Science
Fall 2017
Diane Horton
Function Design Recipe

• CSC108 teaches a “recipe” for writing functions (and methods):
  1. Write examples of calls and the expected returned values.
  2. Write the function header.
  3. Add a description of what the function does.
  4. Write the body.
  5. Test your function.
Type annotations

• We will use Python’s type annotations.
  
```python
def is_even(value: int) -> bool
```

• Different from what some saw in csc108.

• Benefits:
  – Communicates to programmers who will use the function.
  – (They are writing client code.)
  – Allows PyCharm to check that your code conforms.
Design by contract

• Each function (or method) provides a service.
• Its service is specified by the docstring.
• We can think of the docstring as a contract: You call me right and I’ll do the following for you.
• This way of thinking about programming is called Design by contract.