

The Python Language (Part 3)

Copyright © 2026 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - A subset of Python...
 - That is appropriate for COS 333...
 - Through example programs

Agenda

- **Statements**
- Throwing exceptions
- Modules
- Packages
- Object-oriented programming

Statements

- See **euclidclient1.py**

```
$ python euclidclient1.py
Enter the first integer: 8
Enter the second integer: 12
gcd: 4
lcm: 24
$
```

Statements

- See **euclidclient1.py** (cont.)
 - *Unpacking assignment statement*

```
x, y = 1, 2
```

Traditional

```
temp = i%j  
i = j  
j = temp
```

Verbose

```
temp1 = j  
temp2 = i%j  
i = temp1  
j = temp2
```

Python

```
i, j = j, i%j
```

Statements

Assignment statements

var = *expr*

var += *expr*

var -= *expr*

var *= *expr*

var /= *expr*

var // = *expr*

var %= *expr*

var ** = *expr*

var &= *expr*

var |= *expr*

var ^= *expr*

var >> = *expr*

var << = *expr*

Statements

Unpacking assignment statement

```
var1, var2, ... = iterable
```

No-op statement

```
pass
```

`assert` **statement**

```
assert expr, message
```

Statements

Function call statement

```
f(expr, name=expr, ...)
```

return **statement**

```
return
```

```
return expr
```

Statements

```
if statement
    if expr:
        statement(s)
    elif expr:
        statement(s)
    ...
    else:
        statement(s)
```

False, 0, None, '', "", [], (), and {}
indicate logical FALSE
Any other value indicates logical TRUE

Statements

```
while statement  
    while expr:  
        statement(s)
```

False, 0, None, ' ', "", [], (), and {}
indicate logical FALSE
Any other value indicates logical TRUE

Statements

```
for statement  
    for var in iterable:  
        statement(s)
```

```
break statement  
    break
```

```
continue statement  
    continue
```

```
for i in range(0, 5):  
    ... i ...  
for i in range(5):  
    ... i ...
```

Statements

```
try statement
    try:
        statement(s)
    except [ExceptionClass [as var]]:
        statement(s)

raise statement
    raise ExceptionClass(str)
```

Agenda

- Statements
- **Throwing exceptions**
- Modules
- Packages
- Object-oriented programming

Throwing Exceptions

- Recall **euclidclient1.py**

```
$ python euclidclient1.py
Enter the first integer: 0
Enter the second integer: 12
gcd: 12
lcm: 0
$ python euclidclient1.py
Enter the first integer: 0
Enter the second integer: 0
gcd: 0
Traceback (most recent call last):
  File "euclidclient1.py", line 53, in <module>
    main()
  File "euclidclient1.py", line 42, in main
    my_lcm = lcm(i, j)
  File "euclidclient1.py", line 26, in lcm
    return (i // gcd(i, j)) * j
ZeroDivisionError: integer division or modulo by zero
$
```

Throwing Exceptions

- See **euclidclient2.py**

```
$ python euclidclient2.py
Enter the first integer: 8
Enter the second integer: 12
gcd: 4
lcm: 24
$ python euclidclient2.py
Enter the first integer: 0
Enter the second integer: 12
gcd: 12
lcm(i,j) is undefined if i or j is 0
$ python euclidclient2.py
Enter the first integer: 0
Enter the second integer: 0
gcd(i,j) is undefined if i and j are 0
$
```

Throwing Exceptions

```
BaseException
  Exception
    ArithmeticError
      FloatingPointError
      OverflowError (legacy)
    ZeroDivisionError
  AssertionError
  AttributeError
  BufferError
  EOFError
  ImportError
    ModuleNotFoundError
  LookupError
    IndexError
    KeyError
  MemoryError
  NameError
    UnboundLocalError
```

Python
standard
exception
classes

Throwing Exceptions

```
BaseException (cont.)
  Exception (cont.)
    OSError
      BlockingIOError
      ChildProcessError
      ConnectionError
        BrokenPipeError
        ConnectionAbortedError
        ConnectionRefusedError
        ConnectionResetError
      FileExistsError
      FileNotFoundError
      InterruptedError
      IsADirectoryError
      NotADirectoryError
      PermissionError
      ProcessLookupError
      TimeoutError
```

Python
standard
exceptions
(cont.)

Throwing Exceptions

```
BaseException (cont.)
  Exception (cont.)
    ReferenceError
    RuntimeError
      NotImplementedError
      RecursionError
    StopIteration
    StopAsyncIteration
    SyntaxError
      IndentationError
      TabError
    SystemError
    TypeError
    ValueError
      UnicodeError
        UnicodeDecodeError
        UnicodeEncodeError
        UnicodeTranslateError
```

Python
standard
exceptions
(cont.)

Throwing Exceptions

```
BaseException (cont.)  
  Exception (cont.)  
    Warning  
      BytesWarning  
      DeprecationWarning  
      FutureWarning  
      ImportError  
      PendingDeprecationWarning  
      ResourceWarning  
      RuntimeError  
      SyntaxWarning  
      UnicodeWarning  
      UserWarning  
GeneratorExit  
KeyboardInterrupt  
SystemExit
```

Python
standard
exceptions
(cont.)

Agenda

- Statements
- Throwing exceptions
- **Modules**
- Packages
- Object-oriented programming

Modules

- ***Module***
 - A .py file that is designed to be included into a ***client*** .py file

Modules

- Building and running

```
$ python toplevelmodule.py
```

- Automatically compiles/interprets included modules

Modules

- See **euclid.py**, **euclidclient3.py**

```
$ python euclidclient3.py  
Enter the first integer: 8  
Enter the second integer: 12  
gcd: 4  
lcm: 24  
$ python euclidclient3.py  
Enter the first integer: 8  
Enter the second integer: 0  
gcd: 8  
lcm(i,j) is undefined if i or j is 0  
$ python euclidclient3.py  
Enter the first integer: 0  
Enter the second integer: 0  
gcd(i,j) is undefined if i and j are 0  
$
```

Agenda

- Statements
- Throwing exceptions
- Modules
- **Packages**
- Object-oriented programming

Packages

- ***Package***
 - A named group of modules (and other packages)
 - A *module* is stored in a *file*
 - A *package* is stored in a *directory*

Packages

- See **intmath/ __init__ .py**
 - Declares `intmath` as a package
- See **intmath/euclid.py**
 - A module in the `intmath` package

Packages

- See **euclidclient4.py**

```
$ python euclidclient4.py
Enter the first integer: 8
Enter the second integer: 12
gcd: 4
lcm: 24
$ python euclidclient4.py
Enter the first integer: 8
Enter the second integer: 0
gcd: 8
lcm(i,j) is undefined if i or j is 0
$ python euclidclient4.py
Enter the first integer: 0
Enter the second integer: 0
gcd(i,j) is undefined if i and j are 0
$
```

Agenda

- Statements
- Throwing exceptions
- Modules
- Packages
- **Object-oriented programming**

Object-Oriented Programming

- See **fractionprelim.py**,
fractionprelimclient.py

```
$ python fractionprelimclient.py
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
frac1: 1/2
frac2: 3/4
frac1 does not equal frac2
frac1 is less than frac2
frac1 is less than or equal to frac2
-frac1: -1/2
frac1 + frac2: 5/4
frac1 - frac2: -1/4
frac1 * frac2: 3/8
frac1 / frac2: 2/3
$
```

Aside: Name Mangling

- Incidentally:
 - Use of leading **double** underscores causes *name mangling*
 - Example: In Fraction, compiler turns `__num` into `_Fraction__num`

Lecture Summary

- In this lecture we covered these aspects of Python:
 - Throwing exceptions
 - Modules
 - Packages
 - Object-oriented programming
- See also:
 - **Appendix: Duck Typing**

Appendix: Duck Typing

Duck Typing

- Observation:
 - Python uses *duck typing*

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.”

-- James Whitcomb Riley

Duck Typing

- Example: Recall **euclid.py**
 - `i` and `j` parameters of `gcd()` can reference objects of any class, as long as they can be:
 - Operands of `==`
 - Arguments to `abs()`
 - Operands of `!=`
 - Operands of `%`

Duck Typing

Language	Object references (variables, parameters, fields) have types?	Objects (as they exist in memory) have types?	Language classification
C	yes	no	weakly typed
Java	yes	yes	strongly typed
Python	no	yes	dynamically (duck) typed

Duck Typing

- Is duck typing good or bad (vs. strong typing) ?
- See **euclidstrong.py**
 - Which is better, euclid.py or euclidstrong.py?

Duck Typing

- **Style 1:** Don't validate parameter types
 - Validating parameter types is constraining and slow
 - **So euclid.py is better**
- **Style 2:** Validate parameter types
 - Validating parameter types is safe
 - **So euclidstrong.py is better**
- We'll use Style 1

Duck Typing

- Commentary
 - **Small** projects:
 - Maybe need not validate parameter types
 - **Large** projects:
 - Maybe should validate parameter types

Duck Typing

- Commentary
 - But if you feel the need to validate parameter types, then why are you using Python?