The Python Language (Part 2)

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

- **Data types and operators**
- **Terminal I/O**
- Catching exceptions

Data Types and Operators

- See **circle1.py**

```
$ python circle1.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.415927.
$ python circle1.py
Enter the circle's radius:
1
A circle with radius 1 has diameter 2
and circumference 6.283185.
$
```

Why use double quotes instead of single quotes to delimit the `str` literal argument to `input()`?

Data Types and Operators

Data Type	Size	Example Literals
`int`	`4 bytes (no theoretical size limit)`	`1, 23, 3493, 01, 027, 06645, 0x1, 0x17, 0xDA5`
`float`	`8 bytes`	`0., 0.0, .0, 1.0, 1e0, 1.0e0`
`bool`	`1 byte`	`False, True`
`str`	`(varies)`	`'hi',"hi"`

How does Python's `float` compare with Java's `float`?

Data Types and Operators

Conversion Function	Usage
`int(object)`	`frequent`
`float(object)`	`frequent`
`bool(object)`	`infrequent`
`str(object)`	`frequent`

What is the equivalent mechanism in Java?

Data Types and Operators

Operator	(Priority) Meaning
`{key:expr,…}, {expr,…}`	(1) **Dictionary** creation; **set** creation
`[expr,…]`	(2) **List** creation
`(expr,…)`	(3) **Tuple** creation, or just parentheses
`f(expr,…)`	(4) Function call
`x[startindex:stopindex]`	(5) Slicing (for sequences)
`x[index]`	(6) Indexing (for containers)
`x.attr`	(7) Attribute reference
`x**y`	(8) Exponentiation
`~x`	(9) Bitwise NOT
`+x, -x`	(10) Unary plus, unary minus
`x*y, x/y, x//y, x%y`	(11) Mult, div, truncating div, remainder (or string formatting)
`x+y, x-y`	(12) Addition, subtraction

Data Types and Operators

Beware of division operators:

Expression	Value
5 // 2	2
5 / 2	2.5
4 / 2	2.0

In Java, what is the value of `5/2`?

Data Types and Operators

Operator	(Priority) Meaning
`x<<i, x>>i`	(13) Left-shift, right-shift
`x&y`	(14) Bitwise AND
`x^y`	(15) Bitwise XOR
`x|y`	(16) Bitwise OR
`x<y, x<=y, x>y, x>=y`	(17) Relational
`x==y, x!=y`	(17) Relational
`x is y, x is not y`	(18) **Identity** tests
`x in y, x not in y`	(19) **Membership** tests
`not x`	(20) Logical NOT
`x and y`	(21) Logical AND
`x or y`	(22) Logical OR

Terminal I/O

Reading from stdin:

```
str = input()
str = input(prompt_str)

import sys
…
str = sys.stdin.readline()
```

Terminal I/O

Writing to stdout:

```
print(str)
print(str, end='')
print(str1, str2, str3)
print(str1, str2, str3, end='')
```

Writing to stderr:

```
import sys
…
print(str, file=sys.stderr)
print(str, end='', file=sys.stderr)
print(str1, str2, str3, end='', file=sys.stderr)
```

Agenda

- Data types and operators
- Terminal I/O
- **Catching exceptions**

Catching Exceptions

How can you make circle1.py crash? (No fair looking at the next slide.)

Catching Exceptions

- Recall **circle1.py**

```
$ python circle1.py
Enter the circle's radius:
xyz
Traceback (most recent call last):
  File "circle1.py", line 26, in <module>
    main()
  File "circle1.py", line 15, in main
    radius = int(line)
ValueError: invalid literal for int() with base 10: 'xyz'
$ python circle1.py
Enter the circle's radius:
Traceback (most recent call last):
  File "circle1.py", line 26, in <module>
    main()
  File "circle1.py", line 14, in main
    line = input("Enter the circle's radius:\n")
EOFError
$
```

Catching Exceptions

- See **circle2.py**

```
$ python circle2.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.415927.
$ python circle2.py
Enter the circle's radius:
xyz
invalid literal for int() with base 10: 'xyz'
$ python circle2.py
Enter the circle's radius:

$
```

Catching Exceptions

- See **circle3.py**

```
$ python circle3.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.415927.
$ python circle3.py
Enter the circle's radius:
xyz
Error: Not an integer
$ python circle3.py
Enter the circle's radius:
Error: Missing integer
$
```

Catching Exceptions

- See **circle4.py**

```
$ python circle4.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.415927.
$ python circle4.py
Enter the circle's radius:
xyz
Error: Not an integer
$ python circle4.py
Enter the circle's radius:
Error: Missing integer
$
```

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 1:
Python executes *stmt1*, *stmt2, stmt3* successfully

Which statements are executed? (No fair looking at the next slide.)

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 1:
Python executes *stmt1*, *stmt2, stmt3* successfully

Catching Exceptions

```
try:
     stmt1
     stmt2
     stmt3
except ExceptionClass1:
     stmt4
     stmt5
     stmt6
except ExceptionClass2:
     stmt7
     stmt8
     stmt9
stmt10
stmt11
stmt12
```

Case 2:
stmt2 throws an object of some class that matches ExceptionClass1

The thrown object **matches** *ExceptionClass1* if the class of the thrown object is *ExceptionClass1* or any subclass of *ExceptionClass1*

Which statements are executed? (No fair looking at the next slide.)

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 2:
stmt2 throws an object of some class that matches ExceptionClass1

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 3:
stmt2 throws an object of some class that does not match ExceptionClass1, but does match ExceptionClass2

Which statements are executed? (No fair looking at the next slide.)

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 3:
stmt2 throws an object of some class that does not match ExceptionClass1, but does match ExceptionClass2

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 4:
stmt2 throws an object of some class that matches both ExceptionClass1 and ExceptionClass2

Which statements are executed? (No fair looking at the next slide.)

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 4:
stmt2 throws an object of some class that matches both ExceptionClass1 and ExceptionClass2

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 5:
stmt2 throws an object of some class that matches neither ExceptionClass1 nor ExceptionClass2

Which statements are executed? (No fair looking at the next slide.)

Catching Exceptions

```
try:
    stmt1
    stmt2
    stmt3
except ExceptionClass1:
    stmt4
    stmt5
    stmt6
except ExceptionClass2:
    stmt7
    stmt8
    stmt9
stmt10
stmt11
stmt12
```

Case 5:
stmt2 throws an object of some class that matches neither ExceptionClass1 nor ExceptionClass2

Python propagates the exception outward and upward, and repeats the algorithm at each level

Aside: Exit Status

- See **circle5.py**

```
$ python circle5.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.415927.
$ echo $?
0
$ python circle5.py
Enter the circle's radius:
xyz
Error: Not an integer
$ echo $?
1
$ python circle5.py
Enter the circle's radius:
Error: Missing integer
$ echo $?
1
$
```

On MS Windows use:
`echo %errorlevel%`

Lecture Summary

- In this lecture we covered these aspects of Python:
  - Data types and operators
  - Terminal I/O
  - Catching exceptions