

The Python Language (Part 2)

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

- **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 31.42.
Thank you for running the program.
$ python circle1.py
Enter the circle's radius:
1
A circle with radius 1 has diameter 2
and circumference 6.28.
Thank you for running the program.
$
```

Data Types and Operators

- See **circle1options.py**

```
$ python circle1options.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.41592653589793 .
A circle with radius 5 has diameter 10
and circumference 31.41592653589793.
A circle with radius 5 has diameter 10
and circumference 31.42.
A circle with radius 5 has diameter 10
and circumference 31.42.
A circle with radius 5 has diameter 10
and circumference 31.42.
A circle with radius 5 has diameter 10
and circumference 31.42.
A circle with radius 5 has diameter 10
and circumference 31.42.
Thank you for running the program.
$
```

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"

Data Types and Operators

Conversion Function	Usage
<code>int(object)</code>	frequent
<code>float(object)</code>	frequent
<code>bool(object)</code>	infrequent
<code>str(object)</code>	frequent

Data Types and Operators

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

Data Types and Operators

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

Data Types and Operators

Beware of division operators:

Expression	Value
5 // 2	2
5 / 2	2.5
float(5) / float(2)	2.5
4 / 2	2.0
float(4) / float(2)	2.0

Suggestion:
use only
boldfaced
forms

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

- Recall circle1.py

```
$ python circle1.py
Enter the circle's radius:
xyz
Traceback (most recent call last):
  File "circle1.py", line 28, 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:
^d
Traceback (most recent call last):
  File "circle1.py", line 28, in <module>
    main()
  File "circle1.py", line 14, in main
    line = input("Enter the circle's radius:\n")
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
EOFError
$
```

On MS Windows use
^z<Enter>

Catching Exceptions

- See **circle2.py**

```
$ python circle2.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.42.
Thank you for running the program.
$ python circle2.py
Enter the circle's radius:
xyz
invalid literal for int() with base 10: 'xyz'
Thank you for running the program.
$ python circle2.py
Enter the circle's radius:
^d
Thank you for running the program.
$
```

Catching Exceptions

- See **circle3.py**

```
$ python circle3.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.42.
Thank you for running the program.
$ python circle3.py
Enter the circle's radius:
xyz
Error: Not an integer
Thank you for running the program.
$ python circle3.py
Enter the circle's radius:
^d
Error: Missing integer
Thank you for running the program.
$
```

Catching Exceptions

- See **circle4.py**

```
$ python circle4.py
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.42.
Thank you for running the program.
$ python circle4.py
Enter the circle's radius:
xyz
Error: Not an integer
Thank you for running the program.
$ python circle4.py
Enter the circle's radius:
^d
Error: Missing integer
Thank you for running the program.
$
```

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*

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*

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:
^d
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