2. Compound statement structure

header line:

: 
: 
: statement 1
statement 2
: 
: 
: statement n

• No braces or begin/end.
• Indent consistently.
• No statement terminator punctuation.
• Split statements over lines with "\".

2a. Conditional statements

if x < y:
    print("Big!")
else:
    print("small.")
x = x + 1

2b. For loops

for n in [ "how", "are", "you" ]:
    print(n)
# Prints "how", "are", and "you"

for n in range( 5 ):
    print(n)
# Prints 0, 1, 2, 3, and 4

for n in range( 10, 0, -2 ):
    print(n)
# Prints 10, 8, 6, 4, and 2

Python 3.x Summary

3. Classes and Packages

from math import sqrt

class Point( object ):
    "A 2-dimensional point"
    __slots__ = ( "x", "y" )
    def __init__( self, x, y ):
        "constructor"
        self.x = x
        self.y = y
def getX( self ):
    return self.x
def getY( self ):
    return self.y
def distFromOrigin( self ):
    return \sqrt( self.x**2 + \
                 self.y**2 )
def __str__( self ):
    "to-string converter"
    return "(" + \str( self.x ) + \
            "," + \str( self.y ) + ")"

def test():
    p = Point( 3, 4 )
    print(p.getX())
    print(p)
    print(p.distFromOrigin())
test() # Prints 3, "(3,4)", and 5.0

Python 3.x Summary

4. Common functions

int( "52" ) # The integer 52
int( 98.6 ) # The integer 98
str( 52 ) # The string "52"
float( 52 ) # The float 52.0

x = 42
y = 24
print( x )
# Prints 42 on its own line

print( x, y )
# Prints "42 24" on one line

print( str(x) + "|" + str(y) )
# Prints "42|24"

n = int( 
    input( "Number, please: ")
# Reads in literal string;
# int() converts it

(Note: semantics of print and input changed from version 2 of Python.)
2c. While loops

```python
n = 10
while n > 0:
    print(n)
    n = n - 2
# Prints 10, 8, 6, 4, and 2
```

2d. Function definition

```python
def order( str1, str2 ):
    """State which string comes first."
    """
    if str1 < str2:
        print(str1, "comes first")
    else:
        print(str2, "comes first")
def sum3( a, b, c ):
    """Add 3 numbers"
    return a + b + c
```

The string that follows the header is used for documentation generation.

```python
order( "joe", "black" )
# Prints "black comes first"
order( "helga", "smith" )
# Prints "helga comes first"
print( sum3( 1, 5, 9 ) )
# Prints 15
```

5. Built-in data structures

All of the following can be iterated over with a for loop.

String (immutable) – str

Use double or single quotes.
There is no separate character type.
To make a multi-line string, ""
""use 3 double (or single) quotes.""
Indexing with brackets (s[i]) works.

List (mutable; see 1a) – list

```python
x = ["r","o","o","f"]
# works with the str "roof" as well
for i in range( len( x ) ):
    print(x[i])
# Prints "r", "o", "o", and "f"
```

Tuple: an immutable list – tuple

```python
y = ( 4, 5, 6 )
# can't be changed
```

Dictionary/Set (mutable) – dict/set

```python
d = { "fee": 9, "fo": 18 }
# Order of keys is not settable.
d["fum"] = 21
d["fo"] = 17
for key in ("fum","fee","fo"):  
    print(d[key])
# Prints 21, 9, and 17
```

A set is just a dict containing keys without values.

```python
names = {"Manny","Moe","Jack"}
```

1. Basics

Comments begin with "#". Variables are not declared; They can be assigned any type of value at any time.

1a. Data Representation

Everything in Python is an object. Assignment (=) effects sharing of data.

```python
x = [ 1, 2, 3 ]
# a list
y = x
x[ 1 ] = 5  # 2 changed to 5
print(y)  # prints "[1, 5, 3]"
```

Numbers (float, int), bools, and strings can’t be changed; they are for all intents and purposes not shared.

The None object is used to indicate that a variable has no value.

1b. Some operators

Comparison (==, !=, <, <=, >, >=) checks object content, not addresses, for all standard types.

Less common operators

- Logic operators: and, or, not
- String concatenation: +
- Truncating (round-down) division: //
- Normal division: /