Python: Functions and basic data structures

Asokan Pichai
Prabhu Ramachandran

FOSSEE Team

10, October 2009

Outline

1. Python
   - Exercises on Control flow
   - Functions
   - Lists

Problem set 1

- All the problems can be solved using `if` and `while`

Problem 1.1

Write a program that displays all three digit numbers that are equal to the sum of the cubes of their digits. That is, print numbers *abc* that have the property $abc = a^3 + b^3 + c^3$
These are called *Armstrong* numbers.

Problem 1.2 - Collatz sequence

1. Start with an arbitrary (positive) integer.
2. If the number is even, divide by 2; if the number is odd, multiply by 3 and add 1.
3. Repeat the procedure with the new number.
4. It appears that for all starting values there is a cycle of 4, 2, 1 at which the procedure loops.

Write a program that accepts the starting value and prints out the Collatz sequence.

Problem 1.3 - Kaprekar's constant

1. Take a four digit number–with at least two digits different.
2. Arrange the digits in ascending and descending order, giving A and D respectively.
3. Leave leading zeros in A!
4. Subtract A from D.
5. With the result, repeat from step 2.

Write a program to accept a 4-digit number and display the progression to Kaprekar's constant.

Problem 1.4

Write a program that prints the following pyramid on the screen.

```
1
2   2
3   3   3
4   4   4   4
```

The number of lines must be obtained from the user as input.

When can your code fail?

Problem 1.4

Write a program that prints the following pyramid on the screen.

```
1
2  2
3  3  3
4  4  4  4
```

The number of lines must be obtained from the user as input.
When can your code fail? 20 m

Outline

1. Python
   - Exercises on Control flow
   - Functions
   - Lists

Functions: examples

```
def signum( r ):
    """returns 0 if r is zero
    -1 if r is negative
    +1 if r is positive"""
    if r < 0:
        return -1
    elif r > 0:
        return 1
    else:
        return 0
```

Functions: examples

```
def pad( n, size ):
    """pads integer n with spaces
    into a string of length size
    """
    SPACE = ' '
    s = str( n )
    padSize = size - len( s )
    return padSize * SPACE + s
```

What about %3d?

Functions: examples

```
def pad( n, size ):
    """pads integer n with spaces
    into a string of length size
    """
    SPACE = ' '
    s = str( n )
    padSize = size - len( s )
    return padSize * SPACE + s
```

What about %3d?

What does this function do?

```
def what( n ):
    if n < 0: n = -n
    while n > 0:
        if n % 2 == 1:
            return False
        n /= 10
    return True
```

What does this function do?

```
def what( n ):
    i = 1
    while i * i < n:
        i += 1
    return i * i == n, i
```

What does this function do?

```python
def what( n, x ):
    z = 1.0
    if n < 0:
        x = 1.0 / x
        n = -n
    while n > 0:
        if n % 2 == 1:
            z *= x
        n /= 2
        x *= x
    return z
```

Before writing a function

- Builtin functions for various and sundry
- `abs, any, all, len, max, min`
- `pow, range, sum, type`
- Refer here: `http://docs.python.org/library/functions.html`

30 m

Problem set 2

The focus is on writing functions and calling them.

Problem 2.1

Write a function to return the gcd of two numbers.

Problem 2.2

A pythagorean triad (a, b, c) has the property $a^2 + b^2 = c^2$.

By primitive we mean triads that do not 'depend' on others. For example, (4,3,5) is a variant of (3,4,5) and hence is not primitive. And (10,24,26) is easily derived from (5,12,13) and should not be displayed by our program.

Write a program to print primitive pythagorean triads. The program should generate all triads with a, b values in the range 0—100

Problem 2.3

Write a program that generates a list of all four digit numbers that have all their digits even and are perfect squares.
For example, the output should include 6400 but not 8100 (one digit is odd) or 4248 (not a perfect square).

Problem 2.4

The aliquot of a number is defined as: the sum of the *proper* divisors of the number. For example, the aliquot(12) = 1 + 2 + 3 + 4 + 6 = 16.
Write a function that returns the aliquot number of a given number.

Problem 2.5

A pair of numbers (a, b) is said to be amicable if the aliquot number of a is b and the aliquot number of b is a.

Example: `220, 284`

Write a program that prints all five digit amicable pairs. 55 m

Outline

1. Python
   - Exercises on Control flow
   - Functions
   - Lists

List creation and indexing

```
>>> a = [] # An empty list.
>>> a = [1, 2, 3, 4] # More useful.
>>> len(a)
4
>>> a[0] + a[1] + a[2] + a[-1]
10
```

- Indices start with ?
- Negative indices indicate ?

List: slices

- Slicing is a basic operation
- `list[initial:final:step]`
- The step is optional

```
>>> a[1:3] # A slice.
[2, 3]
>>> a[1:-1]
[2, 3]
>>> a[1:] == a[1:-1]
False
```

Explain last result

List: more slices

```
>>> a[0:-1:2] # Notice the step!
[1, 3]
>>> a[::2]
[1, 3]
>>> a[-1::-1]
```

What do you think the last one will do?

Note: Strings also use same indexing and slicing.

List: examples

```
>>> a = [1, 2, 3, 4]
>>> a[:2]
[1, 2]
>>> a[0:-1:2]
[1, 3]
```

Lists are mutable (unlike strings)

```
>>> a[1] = 20
>>> a
[1, 20, 3, 4]
```

List: examples

```
>>> a = [1, 2, 3, 4]
>>> a[:2]
[1, 2]
>>> a[0:-1:2]
[1, 3]
```

Lists are mutable (unlike strings)

```
>>> a[1] = 20
>>> a
[1, 20, 3, 4]
```

Lists are mutable and heterogenous

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a[2] = a[2] + 23
>>> a
['spam', 'eggs', 123, 1234]
>>> a[0:2] = [1, 12] # Replace items
>>> a
[1, 12, 123, 1234]
>>> a[0:2] = [] # Remove items
>>> a.append( 12345 )
>>> a
[123, 1234, 12345]
```

65 m

List methods

```
>>> a = ['spam', 'eggs', 1, 12]
>>> a.reverse() # in situ
>>> a
[12, 1, 'eggs', 'spam']
>>> a.append(['x', 1])
>>> a
[12, 1, 'eggs', 'spam', ['x', 1]]
>>> a.extend([1,2]) # Extend the list.
>>> a.remove( 'spam' )
>>> a
[12, 1, 'eggs', ['x', 1], 1, 2]
```

List containership

```
>>> a = ['cat', 'dog', 'rat', 'croc']
>>> 'dog' in a
True
>>> 'snake' in a
False
>>> 'snake' not in a
True
>>> 'ell' in 'hello world'
True
```

70 m

Tuples: immutable

```
>>> t = (0, 1, 2)
>>> print t[0], t[1], t[2], t[-1]
0 1 2 2
>>> t[0] = 1
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object does not support item
```

- Multiple return values are actually a tuple.
- Exchange is tuple (un)packing

75 m

range() function

```
>>> range(7)
[0, 1, 2, 3, 4, 5, 6]
>>> range( 3, 9)
[3, 4, 5, 6, 7, 8]
>>> range( 4, 17, 3)
[4, 7, 10, 13, 16]
>>> range( 5, 1, -1)
[5, 4, 3, 2]
>>> range( 8, 12, -1)
[]
```

`for...range(...)` idiom

```
In [83]: for i in range(5):
    ....:     print i, i * i
    ....:
    ....:
0 0
1 1
2 4
3 9
4 16
```

for: the list companion

```
In [84]: a = ['a', 'b', 'c']
In [85]: for x in a:
   ....:     print x, chr( ord(x) + 10 )
   ....:
a  k
b  l
c  m
```

Iterating over the list and not the index + reference
what if you want the index?

for: the list companion

```
In [89]: for p, ch in enumerate( a ):
   ....:     print p, ch
   ....:
   ....:
0 a
1 b
2 c
```

Try: `print enumerate(a)` 85 m

Did we meet the goal?

1. Python
   - Exercises on Control flow
   - Functions
   - Lists