CS521 O2
Information Structures with Python

Lecture 3

Guanglan Zhang

guanglan@bu.edu

Some slides adapted from Prof. Eugene Pinsky

Table of Content

Mutability

- There are two types of Python objects
  - ✓ Mutable ones - their values can be modified, e.g. list, set, dict
  - ✓ Immutable ones - their values cannot be modified – primitive types, string, tuple
  - ✓ Custom classes are typically mutable

Function id()

- All objects in Python has its own unique id
- The id is assigned to the object when it is created
- The function is generally used internally in Python
- The id is the object's memory address. It will be different for each time you run the program
- Some objects have same id (actually one object with multiple pointer), like
  - ✓ Small integers between -5 and 256
  - ✓ Small strings (usually less than 20 character)

Mutability of Collections

Collection	Ordered	Mutable
string	yes	no
list	yes	yes
tuple	yes	no
set	no	yes
dictionary	no	yes

Some variations:
Front set (immutable)
Ordered dictionary

Question: Is it possible to modify an immutable object in place?

Constructors – constructing new variables

- A constructor is a special operation used to make a particular object of the type.
  - ✓ int() – return an integer
  - ✓ float() – return a floating-point number
  - ✓ complex() – return a complex number
  - ✓ str() – return a string
  - ✓ list() – return a list
  - ✓ tuple() – return a tuple
  - ✓ dict() – return a dictionary
  - ✓ set() – return a set

- Function type() return the type of an object

Integer Conversion

- Different bases: 2, 8, 10, 16
- A binary number is expressed in the base-2 numeral system, which uses only two digits: 0 and 1.
- The octal numeral system (the base-8 numbers) uses the digits 0 to 7
- The hexadecimal numeral system (the base-16 numbers) use: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

✓ x_int = 30
✓ x_bin = bin(x_int) # base 2, 0b11110
✓ x_oct = oct(x_int) # base 8, 0o36
✓ x_hex = hex(x_int) # base 16, 0x1e

$$\text{x\_int} = 3 \cdot 10^1 + 0 \cdot 10^0$$

$$\text{x\_bin} = 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

$$\text{x\_oct} = 3 \cdot 8^1 + 6 \cdot 8^0$$

$$\text{x\_hex} = 1 \cdot 16^1 + 14 \cdot 16^0$$

Precedence (order) of operations

- From highest precedence to lowest in the below table
- Parentheses are used to override the precedence order and force some operations to be done before others
- Function divmod() combines // and %, return the tuple (x//y, x%y).

Operator	Meaning
()	Parenthesis (grouping)
**	Exponentiation
-x	Negation
*, /, %, //	Multiplication, Division, Remainder, Quotient
+, -	Addition, Subtraction
<, <=, >, >=, !=, ==	Comparison
not x	Boolean NOT
and	Boolean AND
or	Boolean OR

Other operators

- When using augmented assignment, make sure the variable is defined already – you have assigned it a value earlier
- Bitwise logical operators: &, |, <<, >>
- Identity: is, is not
- Membership: in, not in
- many operators are polymorphic

Assignment Operator	Example
=	x = 1
+=	x += 1 same as x = x+1
-=	x -= 1 same as x = x-1
*=	x *= 2 same as x = x*2
/=	x /= 2 same as x = x/2
**=	x **= 2 same as x = x**2
//=	x //= 2 same as x = x//2

Iteration and Indexing

- We can use indexing to refer individual items within a string, list, and tuple by position
- Objects are “zero-indexed”
- Negative Indexing: We can index from the opposite end using negative integers.
- We can iterate over collections, such as strings, lists and tuples in multiple ways:
  - ✓ Using an explicit index
  - ✓ Use range() object
  - ✓ Use enumerate() function

0	1	2	3	4	5
P	y	t	h	o	n
-6	-5	-4	-3	-2	-1

Iteration using an explicit index

Python 3.6
(known limitations)

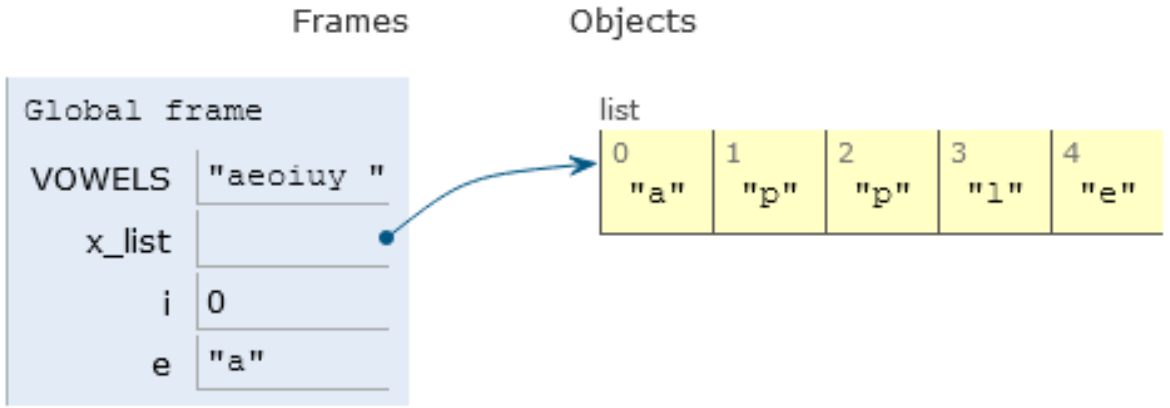
```
1  VOWELS = 'aeoiuy '
2  x_list = ['a','p','p','l','e']
3
4  i = 0
5  while i < len( x_list ):
6      e = x_list [i]
7      if e in VOWELS :
8          print (e,i)
9      i = i + 1
```

Edit this code

executed
execute

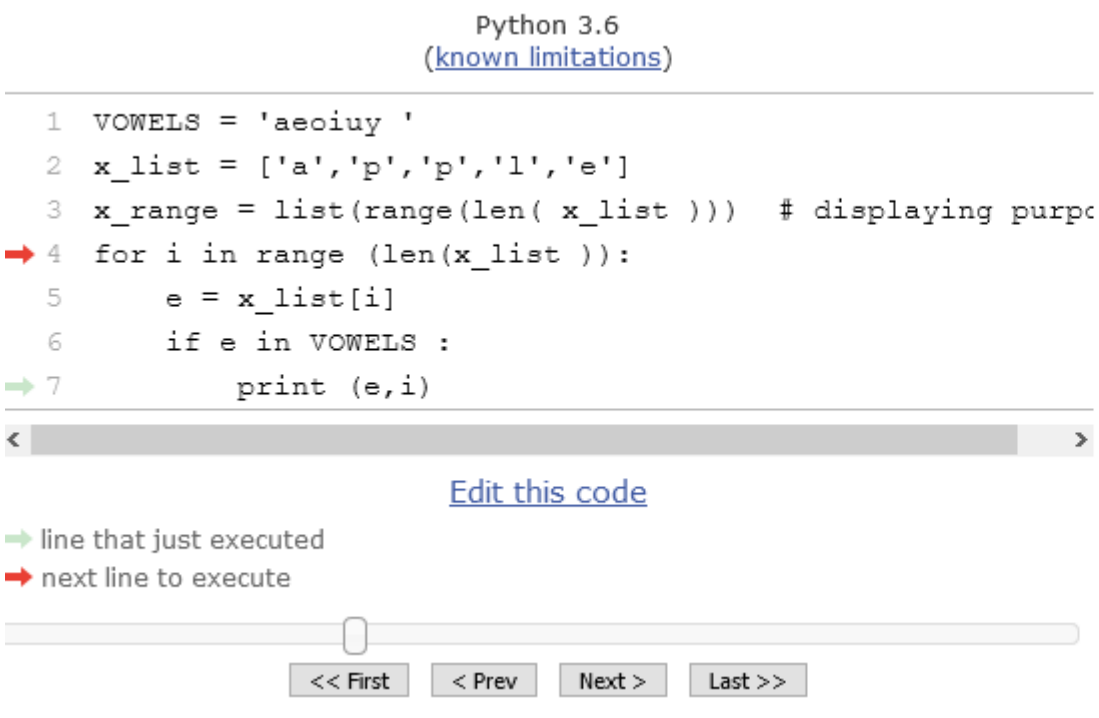
Print output (drag lower right corner to resize)

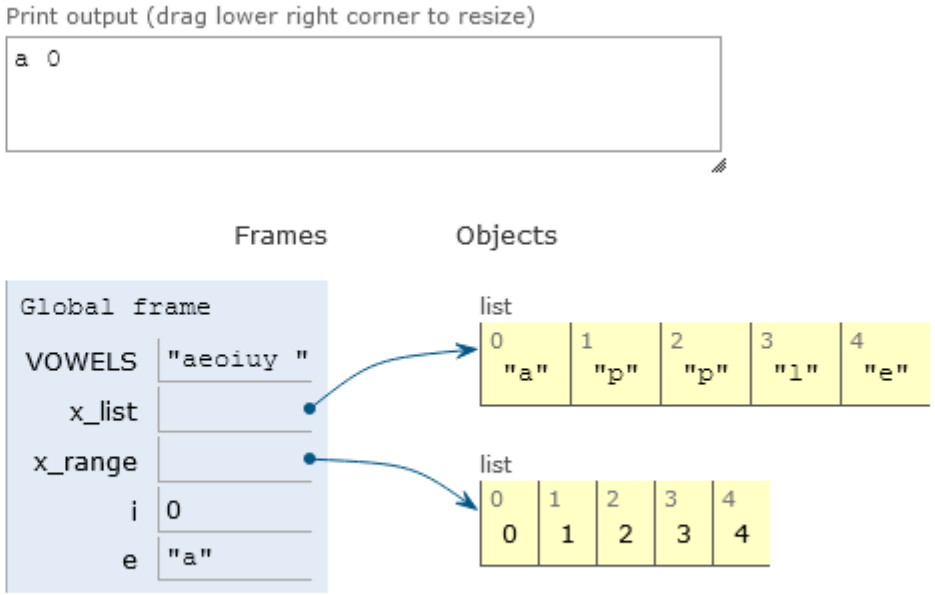
```
a 0
```



Iteration using range()

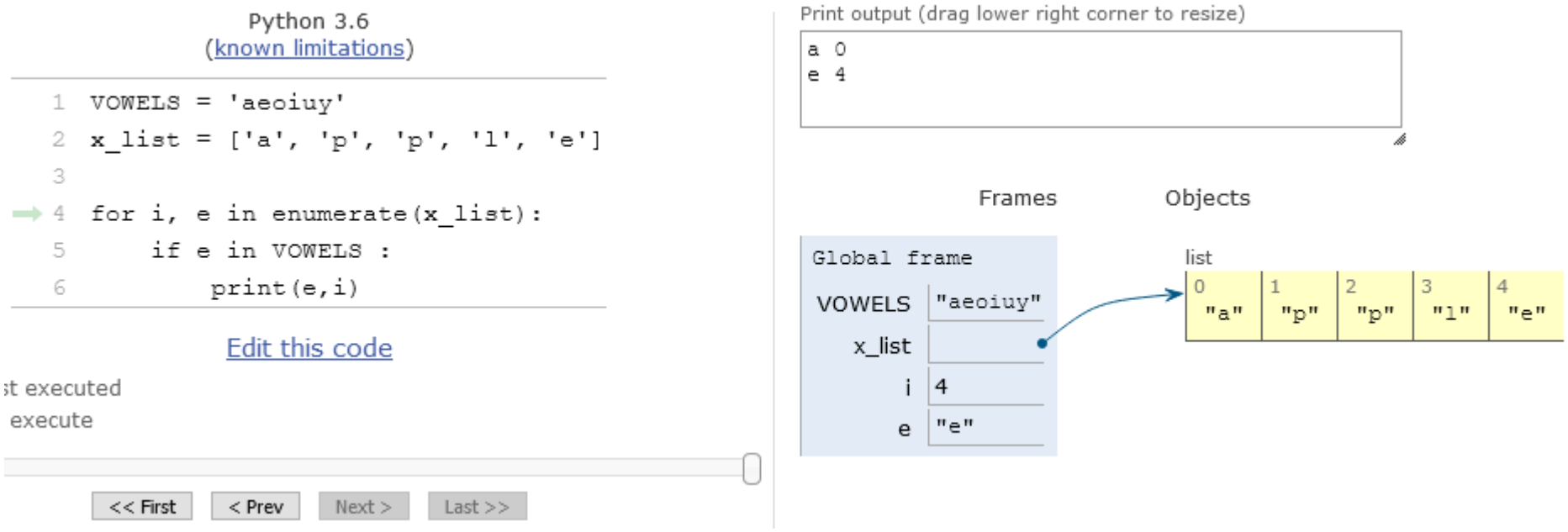
range() returns an object that produces a sequence of integers from start (inclusive) to stop (exclusive) by step - range(i, j) produces i, i+1, i+2, ..., j-1.





Iteration using enumerate()

- enumerate() yields pairs containing an index (from start, which defaults to zero) and the corresponding value
- Can only be used on ordered collections, such as list, string, tuple



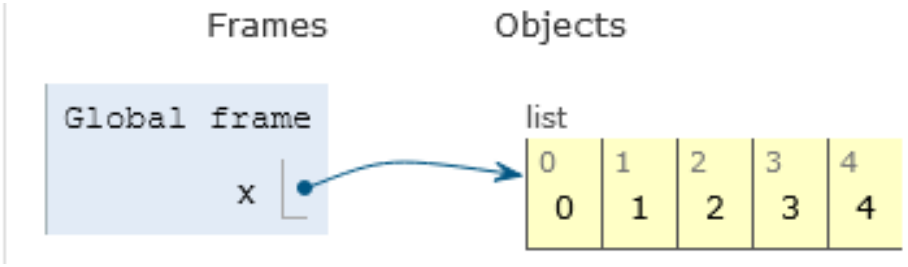
Range object

- range() returns an object that produces a sequence of integers from start (inclusive) to stop (exclusive) by step - range(i, j) produces i, i+1, i+2, ..., j-1.
- Only the stop parameter is required. Start defaults to 0.
- The step parameter specifies the numeric distance between each integer generated in the range sequence. Default step value of 1.

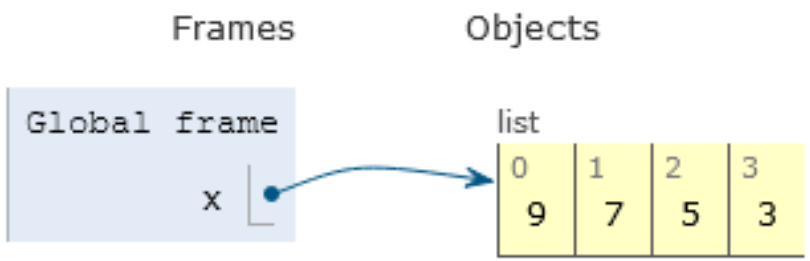
- Two constructions:
  - ✓ range(stop)
  - ✓ Range([start], stop[, step])

Examples:

x = list(range(5))



x = list(range(9, 2, -2))



Example using range()

Calculate the sum of first n positive even integers

Python 3.6
(known limitations)

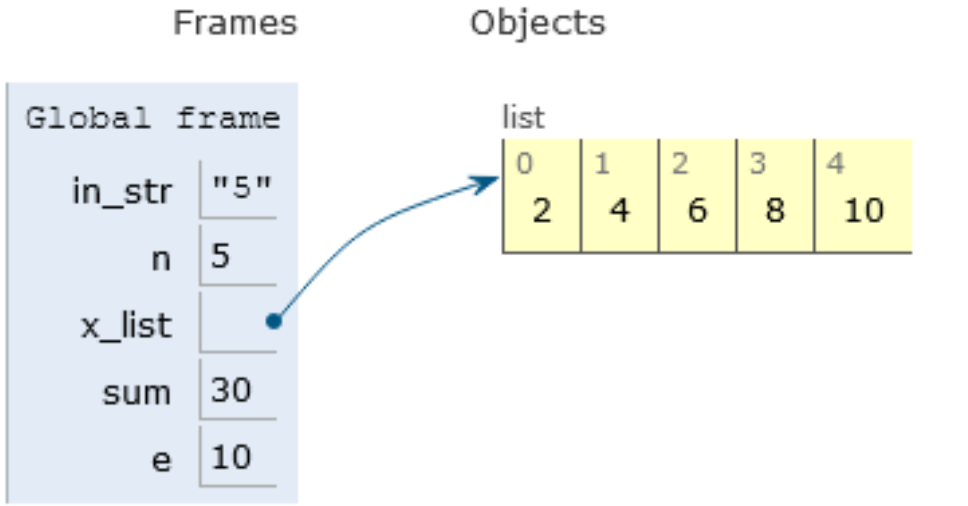
```
1  in_str = input("Enter a positive integer:")
2  n = int(in_str)
3  x_list = list(range(2, 2*n+1, 2))
4  sum = 0
5  for e in x_list:
6      sum += e
7  print(n, sum)
```

Edit this code

:hat just executed
line to execute

Print output (drag lower right corner to resize)

```
Enter a positive integer:5
5 30
```



Exercises

Which built-in types are immutable?

Write a program that constructs a list of every 7-th integer from 1 to 50

Write a program that sums up every 7-th integer from 1 to 50

Key takeaways

- list, set, and dict are mutable (dictionary keys are immutable)
- Primitive data types, string, and tuple are immutable
- Custom classes are typically mutable
- Python objects are “zero-indexed”
- Negative Indexing: We can index from the opposite end using negative integers
- We can iterate over collections in multiple ways:
  - ✓ Using an explicit index
  - ✓ Use range() object
  - ✓ Use enumerate() function