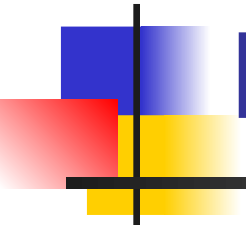
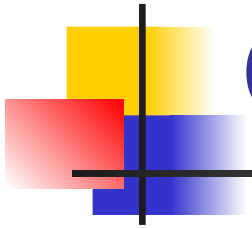


Python Coding

Dr. Bernard Chen

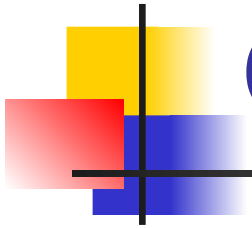
Troy University



Outline

- Intro
- Numbers
- Strings
- Lists

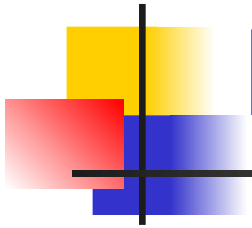
Introduction to Computer Science



- What do Computing Professionals Do?
  - We write program
  - Developing compute applications that address a need in some activity we humans do.

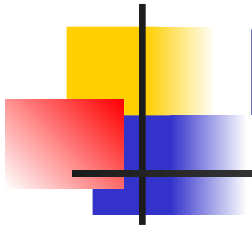
For example

Activity	Computer Application
Game	
Driving	GPS-based navigation software
Bank	Financial tools for investment
Shopping	Recommender system that suggest products
Films	3D computer graphics



Program

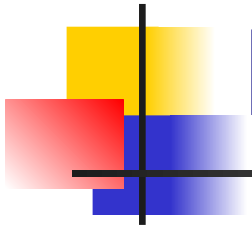
- What distinguishes computer and other machines is that computers can be programmed
- The instructions that are actually executed inside computer is using binary notation (a sequence of 0s and 1s)
- There are many programming languages out there. Such as C++, C#, JAVA, Perl, and Python.



Python

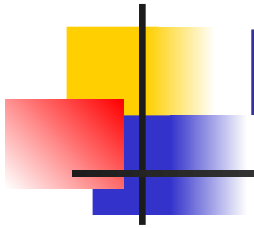
- In this semester, we introduce Python programming language and use it to illustrate core computer science concepts and learn programming

Why do people use Python?



- Software Quality
- Developer productivity
- Program portability
- Support Libraries
- Component integration

Why do people use Python?



- **Software Quality:**

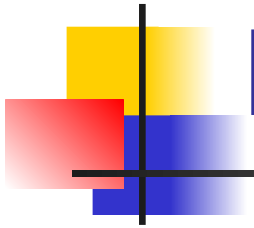
  Python is designed to be readable, and hence maintainable.

  Python is deep support for software reuse mechanisms such as OO.

- **Developer productivity:**

  Python code is typically 1/3 to 1/5 the size of equivalent C++ or JAVA code

Why do people use Python?

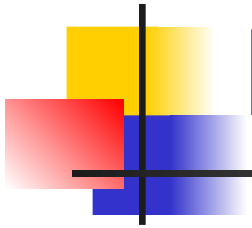


- Program portability

  Most python programs run unchanged on all major computer platforms
- Support Libraries
- Component integration

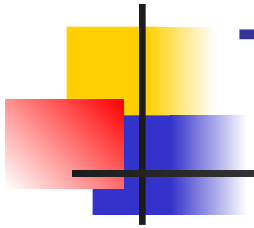
  Today, Python code can invoke C and C++ libraries, can be called from C and C++, can integrate with Java components. Can communicate over XML, Corba and .NET etc

What can I do with Python?

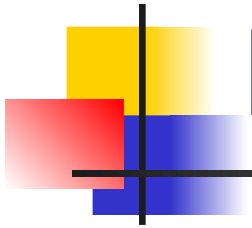


- System Programming
- GUIs
- Internet Scripting
- Database Programming
- Games, Images, AI, XML and more

What are Python's Technical Strength

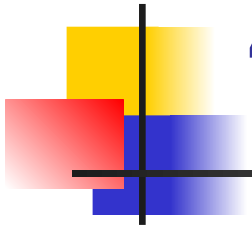


- It's OO (Object Oriented)
- It's free
- It's Portable
- It's Powerful
- **It's Easy to use**
- **It's Easy to learn**



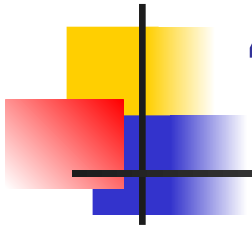
Hello World Program

- Implement by three different methods



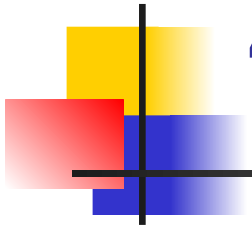
“Hello World” in C

```
main()
{
  printf("hello, world!\n");
}
```



“Hello World” in JAVA

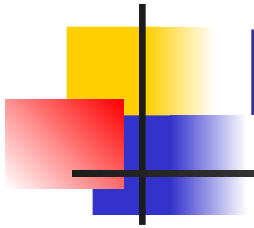
```
class myfirstjavaprog
{
  public static void main(String args[])
  {
    System.out.println("Hello World!");
  }
}
```



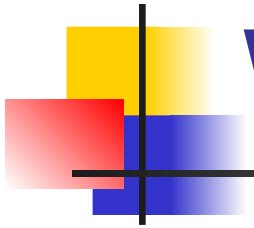
“Hello World” in Python

```
print ("hello World!“)
```

What is the Downside of Python?

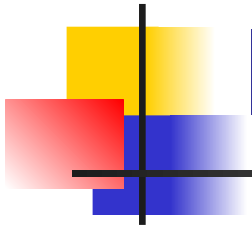


- Perhaps the only downside to Python is that the execution speed may not always as fast as compiled languages such as C and C++
- Python is not compiled all the way down to binary machine code, it compiled to byte code instead.



Who Uses Python Today?

- Google and Yahoo currently use Python in Internet service
- IBM use Python for hardware testing
- Industrial Light and Magic use Python in the production of movie animation
- For more details, visit www.python.org



Install Python

- Go to https://www.python.org/downloads/

  and download the latest version of

  Python

Download Python | Python.org

python.org/downloads/

YouTube 首頁 - Netflix

Python | PSF | Docs | PyPI | Jobs | Community

python™

Donate | Search | GO | Socialize

About | Downloads | Documentation | Community | Success Stories | News | Events

Download the latest version for Windows

Download Python 3.13.1

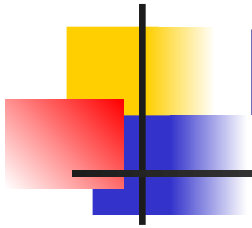
Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.14? Pre-releases, Docker images

Active Python Releases

For more information visit the Python Developer's Guide.

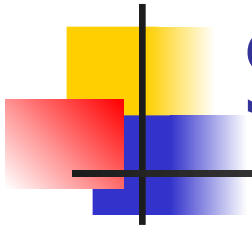
Python version	Maintenance status	First released	End of support	Release schedule
3.14	pre-release	2025-10-01 (planned)	2030-10	PEP 745
3.13	bugfix	2024-10-07	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693



How do you run programs?

Three different methods:

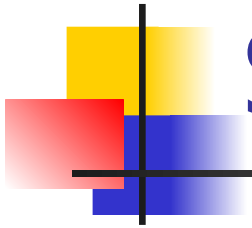
- 1. Interactive Coding
- 2. Files (such as NotePad, WordPad)
- 3. Integrated Development Environment (IDE)



Some more Python codes

- >>> 6+9
- >>>6-9
- >>> 3/4
- >>> 3/4.0
- >>> 3**4

Some more Python codes

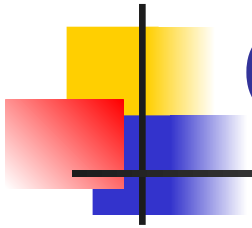


- >>> print “Hello World!”
- >>> print 6+9
- >>> print “6+9”

How do you run programs?

- Code in IDE

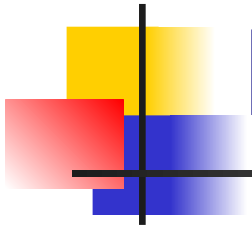
- Step 1. Select “new” in File
- Step 2. Select “Python Script” then click OK
- Step 3. Type in print "Hello World!“
- Step 4. Select “Save” in file (as “.py”)
- Step 5. Select “Run” (F5) in file



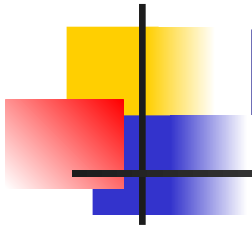
Outline

- Intro
- Numbers
- Strings
- Lists

Python Build-in types



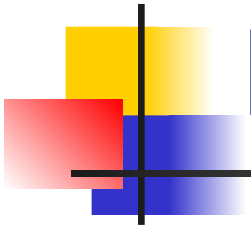
- Numbers 3.1415
- Strings “Hello World”
- Lists [1,2,3,4]
- Files
  input=open(‘file.txt’, ‘r’)



Numbers

Many different types, we only provide 3 here for now:

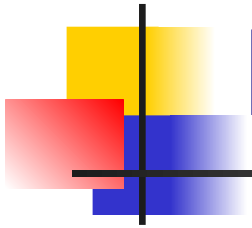
- Integer
- Floating-point
- Long integer



Number in Action

- Perhaps the best way to understand numerical objects is to see them in action.

```
>>> a=3     #Name Created
>>> b=4
```



Number in Action

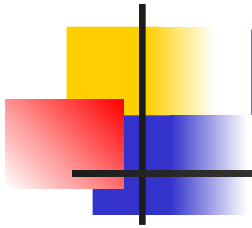
```
>>> a+1  # (3+1)
4

>>> a-1   #(3-1)
2
```

Number in Action

```
>>>  b*3
12

>>>b/2
2
```

Number in Action

```
>>>  b**2    #power
16

>>> a%2 #remainder
1
```



Number in Action

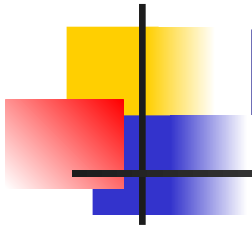
```
>>>  2+4.0, #mix type
6.0

>>>  2.0**b
16.0
```

Number in Action

```
>>>  b / 2 + a
```

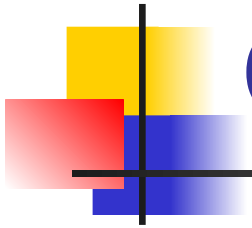
```
>>>  b/(2.0+a)
```

```
>>>  b/(2+a)
```



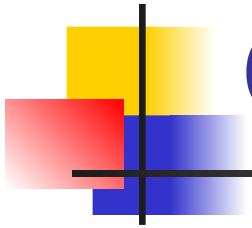
Number in Action

```
>>>  1 / 2.0
0.5

>>> 1/2
0.5

>>> int(1/2)
0
```



Other Numeric Tools

```
>>>int(2.567)
2

>>> round(2.567)
3.0

>>> round(2.567, 2)
2.57
```

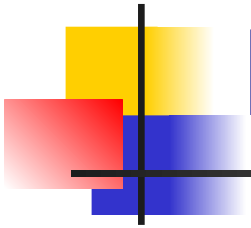


Other Numeric Tools

Math library

```
>>> import math
>>> math.pi
3.14159265358979 3

>>> math.e
2.71828182845904 5
```

(in case you are interesting: http://docs.python.org/library/math.html)

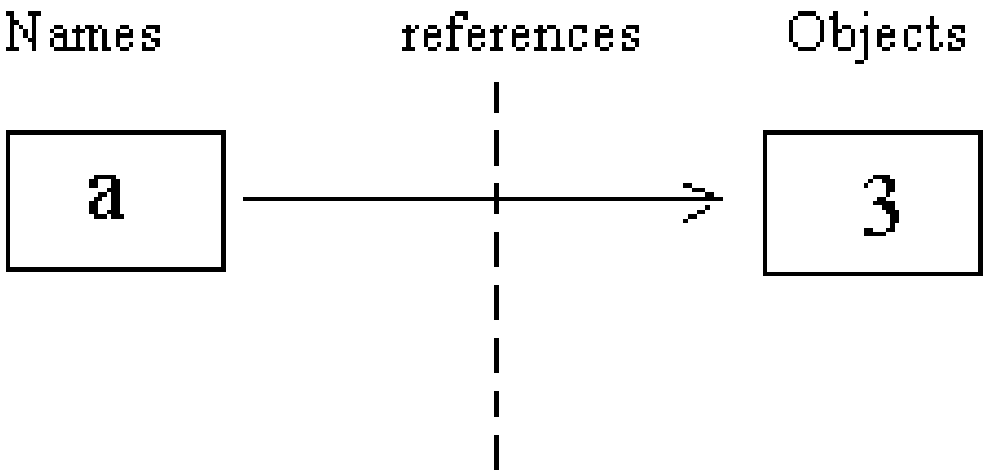


Dynamic Typing

```
>>> a = 3
```

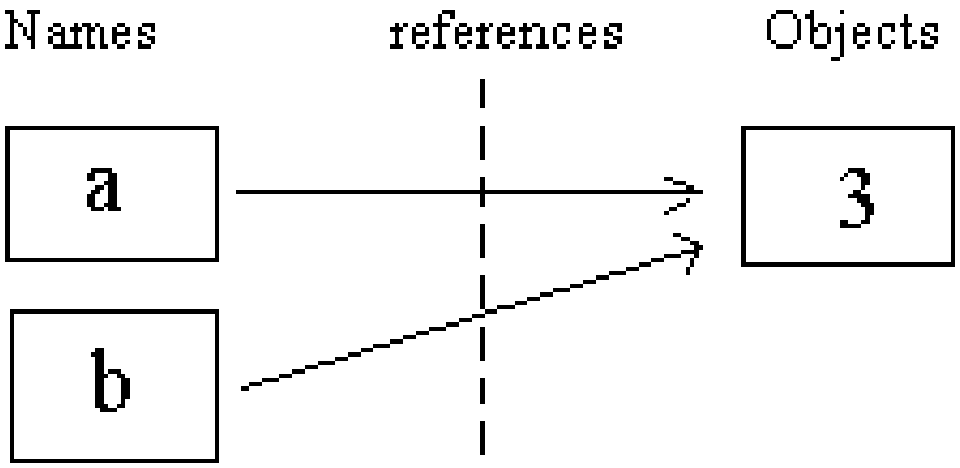
1. Create an object to represent the value of 3
2. Create the variable a, if it does not yet exist
3. Link the variable a to the new object 3

a = 3



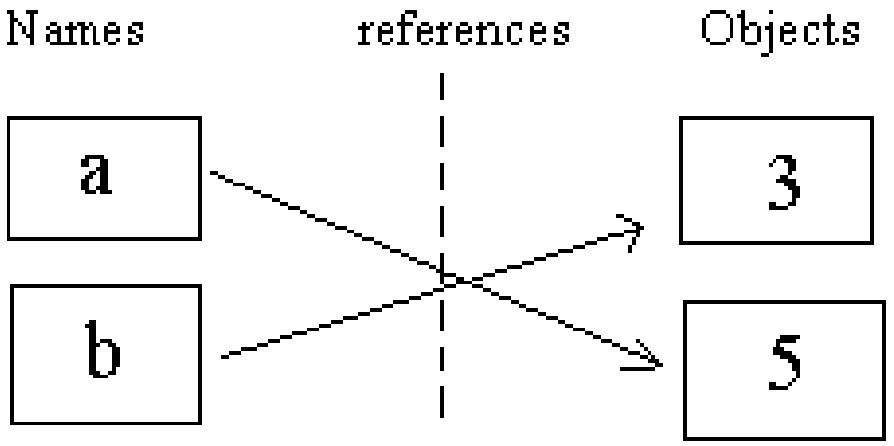
Share Reference

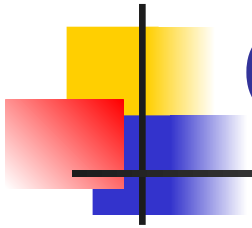
```
>>> a=3
>>> b=a
```



Share Reference

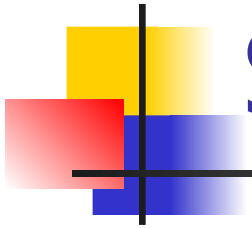
```
>>>a=3
>>>b=a
>>>a=5
```





Outline

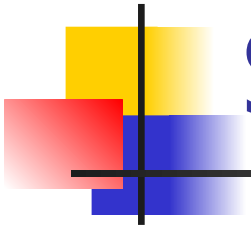
- Intro
- Numbers
- Strings
- Lists



Strings

- The next major build-in type is the Python STRING --- an **ordered** collection of characters to *store* and *represent* text-based information

- Python strings are categorized as *immutable sequences* --- meaning they have a **left-to-right order** (sequence) and cannot be changed in place (immutable)



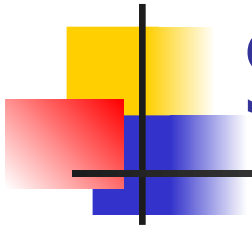
Single- and Double-Quoted

- Single- and Double-Quoted strings are the same

>>> ‘Hello World’ , “Hello World”

- The reason for including both is that it allows you to embed a quote character of the other inside a string

>>> “knight’s” , ‘knight”s’



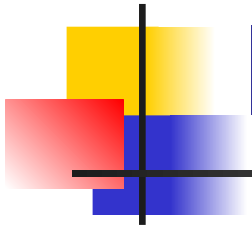
Strings in Action

- Basic operations:

1. len()

2. +

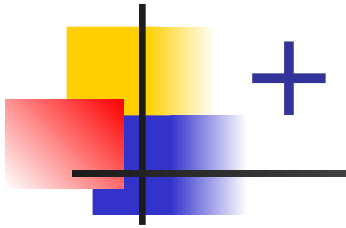
3. *



len()

- The len build-in function returns the length of strings

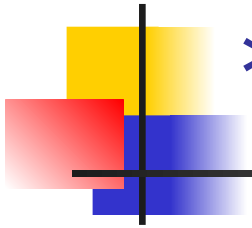
```
>>> len('abc')

>>> a='abc'
>>> len(a)
```



+

- Adding two string objects creates a new string object

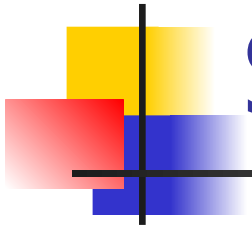
```
>>> ‘abc’ + ‘def’

>>> a=‘Hello’
>>> b=‘World’
>>> a + b
>>> a+ ‘ ’ +b
```



*

- Repetition may seem a bit obscure at first, but it comes in handy in a surprising number of contexts
- For example, to print a line of 80 dashes

```
>>> print ('_' * 80)
```



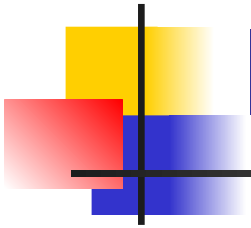
Strings in Action

- Advanced Operations

1. Indexing
2. Slicing
3. String Conversion
4. String Replace

Indexing

- S = ‘STRINGINPYTHON’

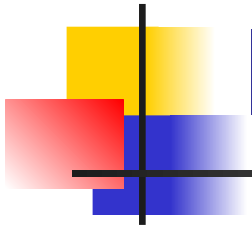


Indexing

- As in the C language, Python offsests start at zero and end at one less than the length of the string

- The first item is at offset 0
- S[0] fetches the item at offsest 0 from the left

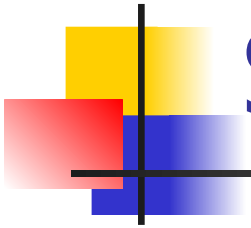
```
>>> S[0], S[5]
```



Index

- Negative indexes mean to count backwards from the end
- S[-2] gets the item offset 2 from the end

```
>>> S[-2]
```



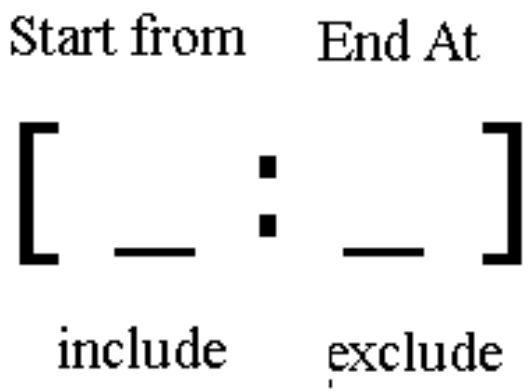
Slicing

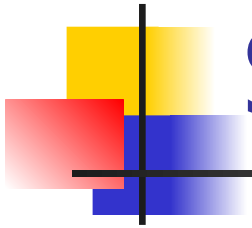
- Slicing allows us to extract an entire section (substring) in a single step

```
>>> S[1:3]   #extract item at 1 and 2
>>> S[1:]    #all items pass the first
>>> S[:-1]   #fetch all but the last item
```

Slicing

- You can regard the slicing function as

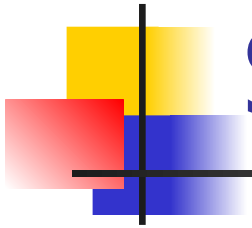




Slicing

- S[1:3]
- S[1:]
- S[:3]
- S[:-1]
- S[:]

String Conversion



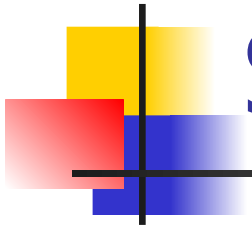
- You cannot add a number and a string together

>>> “42” + 1

- You can use int() function to convert string into integer

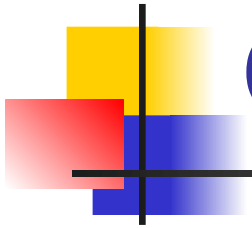
>>> int(“42”)

String Conversion



- Therefore, you can force adding one string to another number, and vise versa

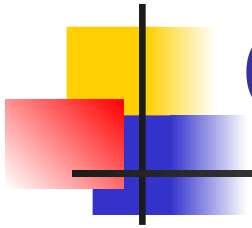
```
>>> int(“42”) +1
>>> “42” + str(1)
```



Changing Strings

- Remember the term “immutable”?
- It means that you cannot change a string in-place

```
>>> S = ‘Spam’
>>> S[0] = ‘X’
```

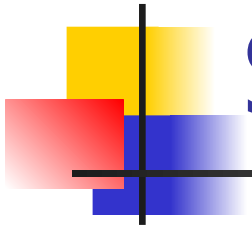


Changing Strings

- There’s a function called replace()

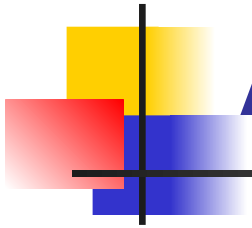
```
>>> S= ‘string in python’
>>> s.replace( ‘string’, ‘number’)
```

Escape Sequence Code Special Bytes



- \n --- Newline (table 5-2)
- \t --- Horizontal Tab

```
>>>  s='a\nb\tc'
>>>  s
'a\nb\c'
>>>  print s
a
b        c
```



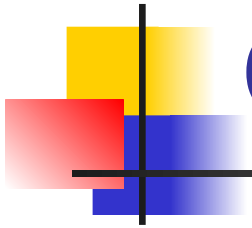
Assignment

- In the interactive shell, define:

```
>>> s1='-'
>>> s2='+'
```

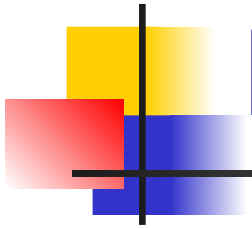
- Now, use s1 and s2 and string operand + and * to present:

1. '-+'
2. '+--'
3. '+--+--'
4. '+-+++--+-+++--+-+++--+-+++--' (do this one)



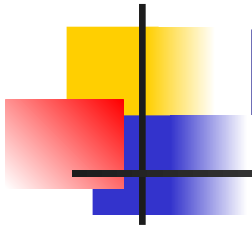
Outline

- Intro
- Numbers
- Strings
- Lists



Lists

- Lists are Python's most flexible **ordered** collection object type
- Lists can contain any sort of object: numbers, strings and even other lists

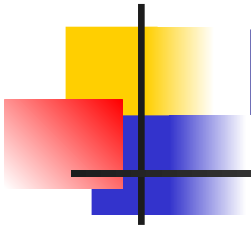


Lists

- **Ordered collections of arbitrary objects**

--- from the functional view, lists are just a place to collect other objects

--- Lists also define a left to right positional ordering of the items in the list

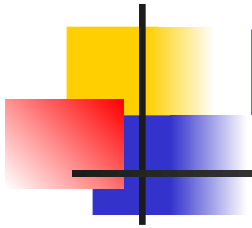


Lists

- **Accessed by offset**

--- Just as strings, you can fetch a component object out of a list by indexing the list on the object's offset

--- you can also for such tasks as slicing and concatenation

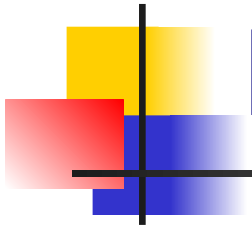


Lists

- **Variable length, heterogeneous, arbitrary nestable**

--- Unlike strings, list can grow and shrink in place

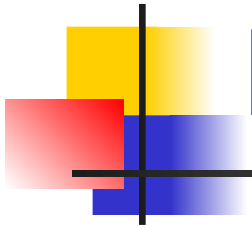
--- lists may contain any sort of object, not just one-character strings (they are heterogeneous)



Lists

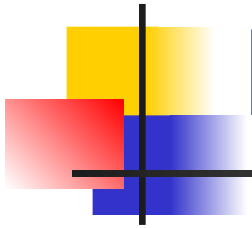
- >>> aa=[] #An empty list
- >>> aa=[1,2,3,4] #4 items, index 0-3

List in action



- List respond to the + and * operations much like strings

```
>>> aa=[1,2,3]
>>> bb=[4,5,6]
>>> aa+bb


>>> aa*3
```

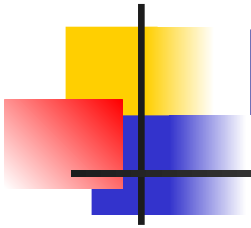


List in action

- Lists also have the function len() to tell the size of lists and “in” function

```
>>> aa=[1,2,3]
>>> len(aa)
```

Indexing, Slicing, and Matrixes

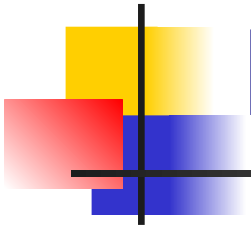


- Besides lists are sequences, index and slicing work the same way

```
>>> aa=[1,2,3]
>>> aa[2]
>>> aa[-2]
>>> aa[1:]
```

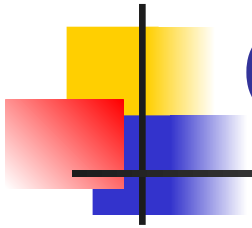
(also try some strings in aa)

Indexing, Slicing, and Matrixes



- We can also assign lists in a list

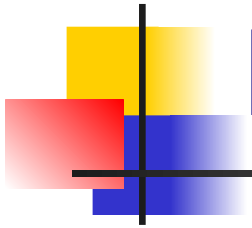
```
>>> matrix=[[1,2,3],[4,5,6],[7,8,9]]
>>> matrix[0]
>>> matrix[1][0]
>>> matrix[0][2]
```



Changing Lists in-place

- When using a list, you can change its contains b assigning to a particular item (offset), or an entire section (slice)

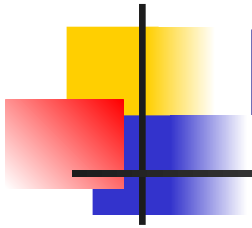
```
>>> aa=[1,2,3]
>>> aa[0]=4
>>> aa[1:]=[5,6]
```



List method calls

The list append method simply tacks a single item onto the end of the list

```
>>> aa=[]
>>> aa.append(1)
>>> aa.append(2)
>>> aa.append(3)
>>> aa.append('4')
>>> aa.append(5.0)
```

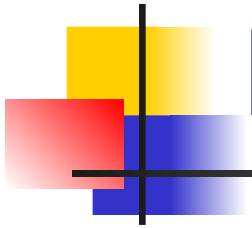


List method calls

- The sort function orders a list in place (in ascending fashion)

```
>>> aa=[4,2,6,8,1,3,4,10]
>>> aa.sort()
```

(also try strings in lists)

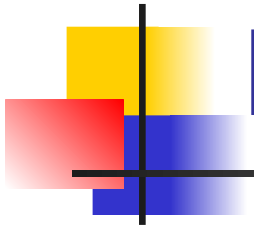


List method calls

- ‘reverse’ reverse the list in-place

```
>>> aa=[1,2,3,4]
>>> aa.reverse()
```

List method calls



- ‘del’ can used to delete an item or section

```
>>> aa=[1,2,3,4,5,6]
>>> del aa[0]
>>> del aa[2:]
```