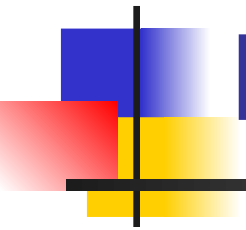
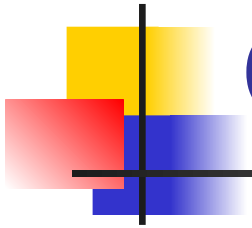


Python Coding part2

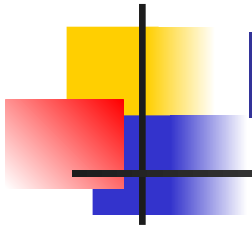
Dr. Bernard Chen

Troy University



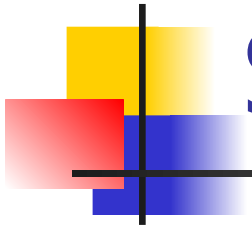
Overall Outline

- If Statement
- Loop
- Function



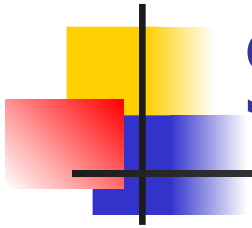
If Statement Outline

- Logic expression
- General concept behind the Python statement syntax
- IF statement



Some Logic Expression

- Computer only understand ‘1’ and ‘0’
- In computer logic:
  - 1: true
  - 0: false



Some Logic Expression

Equal: “==”

NOT Equal: “!=”

Greater: “>”, “>=”

Less than: “<”, “<=”

Some Logic Expression example

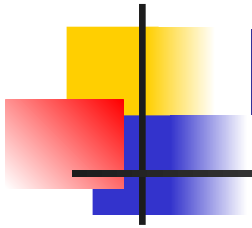
```
>>> a=10
>>> b=10
>>> a==b

>>> a=10
>>> b=5
>>> a==b
```

Some Logic Expression example

```
>>> a=10
>>> b=10
>>> a!=b

>>> a=10
>>> b=5
>>> a!=b
```

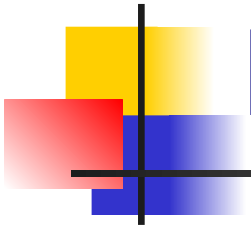
Some Logic Expression example

```
>>> a=10
>>> b=10
>>> a>=b

>>> a=10
>>> b=5
>>> a<b
```



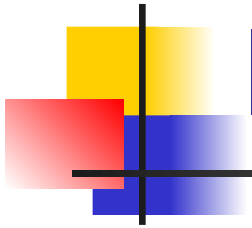
If Statement Outline

- Logic expression
- General concept behind the Python statement syntax
- IF statement



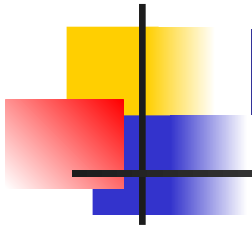
Python's Syntax

- In general, Python has a simple syntax:
  - Block and statement boundaries are detected automatically: python use indention of statement under a header to group the statement
  - Header --- ':'



If Statement Outline

- Logic expression
- General concept behind the Python statement syntax
- IF statement
  - Simple If statement
  - If... else statement
  - If... else if ... else statement

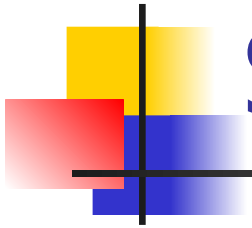


If Statement

- The main statement used for selecting from alternative actions based on test results

- It’s the primary selection tool in Python and represents the ***Logic*** process

Simple If statement



- It takes the form of an if test

- if <test>:

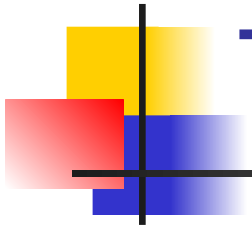
  <statement>

Simple If statement

- if 4>3:
  print (“it is true that 4>3”)

- if 1:
  print (“true”)

- a=10
- if a==10:
  print (“a=10”)



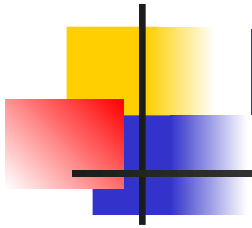
Truth Statement

- In Python:
  - True means any nonzero number or nonempty object
  - False means not true: a zero number of empty object

If Statement “login” example

```
>>> password="Spring 2026"
>>> if password==input("please input your password: "):
...     print("You have successfully login")
...
...
please input your password: Spring 2026
You have successfully login
>>>
>>>
>>>
>>> if password==input("please input your password: "):
...     print("You have successfully login")
...
...
please input your password: spring 2026
>>>
```

If... else statement



- It takes the form of an if test, and ends with an optional else block

- if <test>:

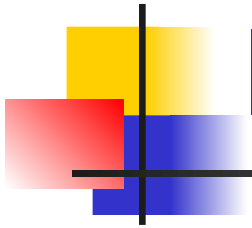
    <statement1>

  else:

    <statement2>

If Statement “login” example

```
>>> if password==input("please input your password: "):
...     print("You have successfully login")
... else:
...     print("Incorrect Password")
...
...
please input your password: spring
Incorrect Password
>>>
```



If... else statement

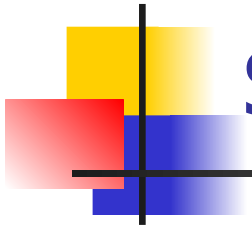
```
>>> a=10
>>> b=5
>>> if a>b:
        print (“a is larger”)
      else:
        print (“b is larger”)
```

If... else if ... else statement

- It takes the form of an if test, followed by one or more optional elif tests, and ends with an optional else block

```
if <test1>:
     <statement1>
elif <test2>:
     <statement2>
else:
      <statement3>
```

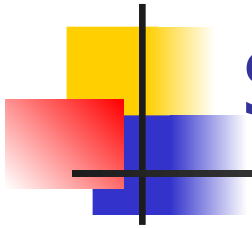
If... else if ... else statement



- a=10
- if a>10:

```
    print (“a > 10”)
elif a==10:
    print (“a = 10”)
else:
    print (“a < 10”)
```

If... else if ... else statement example



```
number = 23
guess = int(input('Enter an integer : '))

if guess == number:
    print ("Congratulations, you guessed it.")
    print ("but you do not win any prizes!")
elif guess < number:
    print ('No, it is a little higher than that')
else:
    print ('No, it is a little lower than that')
```

If... else if ... else statement example

```
import random

number = random.randint(0, 100)
guess = int(input('Enter an integer : '))

if guess == number:
    print ("Congratulations, you guessed it.")
    print ("but you do not win any prizes!")
elif guess < number:
    print ('No, it is a little higher than that')
else:
    print ('No, it is a little lower than that')
```

Some More Logic Expression

- And

(True if both are true)

Examples:

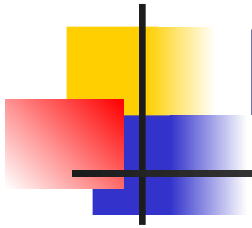
>>> 5>4 and 8>7

True

>>> 5>4 and 8>9

False

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1



Example

```
temp=int(input("Please enter today's tempeture:"))
rain=int(input("Is is raining? 1: yes, 0:no"))

if temp>86 and rain==0:
    print ("Today is a hot sunny day")
elif temp>86 and rain==1:
    print ("Today is a hot raining day")
elif temp<32 and rain==0:
    print ("Today is a cold sunny day")
elif temp<32 and rain==1:
    print ("Today is a cold raining day")
else:
    print ("Today's weather is not special")
```

Some More Logic Expression

- Or

(True if either one is true)

Examples:

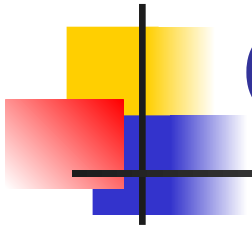
>>> 5>4 or 8>7

True

>>> 5>4 or 8>9

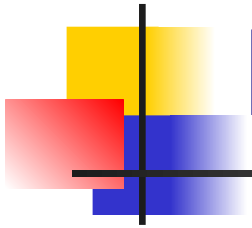
True

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1



Overall Outline

- If Statement
- Loop
- Function

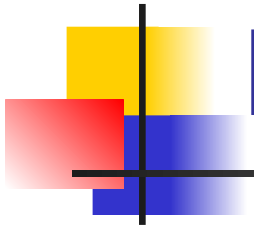


For loop

- The for loop is a generic sequence iterator in Python

- It can step through the items in **ANY <u>ordered</u>** sequence object

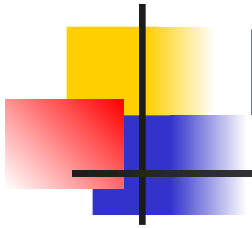
For loop



- The Python for loop begins with a header line that specifies an **assignment target** along with an **object** that you want to step through

```
for <target> in <object>:
        <statement>
```

- When Python runs a for loop, it assigns item in the sequence object to the **target** “one by one”, and then executes the loop body

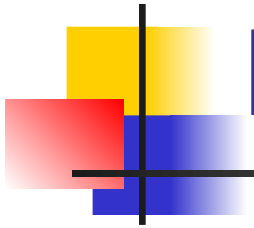


For loop

- The name used as the assignment target in a for header line is usually a **variable** in the scope where the for statement is coded

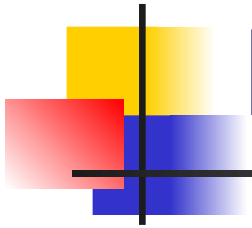
- After the loop, this variable normally still refers to the last item visited

For loop examples



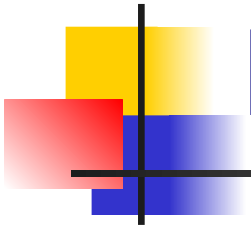
- The name x is assigned to each of the three items in the list in turn, from left to right

```
>>> aa=[“spam”, “eggs”, “ham”]
>>> for i in aa:
        print i
```



For loop examples

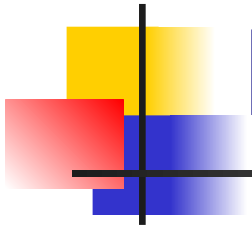
```
>>> aa=[1,2,3,4]
>>> sum=0
>>> for i in aa:
                sum = sum + i
>>> product=1
>>> for i in aa:
                product = product * i
>>> sum, product
```



For loop examples

- The name x is assigned to each of the three items in the list in turn, from left to right

```
>>> s="string in python"
>>> for i in s:
        print i
```



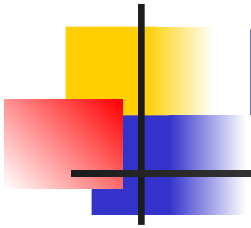
For loop examples

- Several students take the Quiz, please write a program to calculate their average score.

For loop examples

- How many ‘i’ in string s=“string in python”

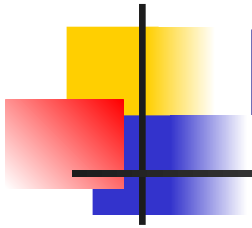
```
>>> s='string in Python‘
>>> count=0
>>> for i in s:
                    if i=='i':
                                count=count+1
>>> count
2
```



For loop examples

- Implement “Factorial” that takes a user input nonnegative integer and returns its factorial

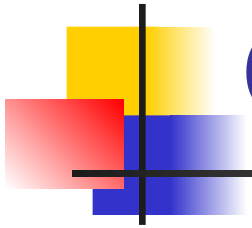
- n!= n*(n-1)*(n-2)*…*2*1 if n>0
- n!=1 if n=0



Loop Variations

- There are also situations where you will need to iterate in a more specialized way in the loop operation.

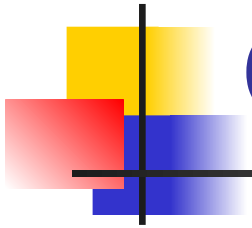
- For example, what if you need to visit every second or third item in a list?
- We have “range” function to help



Counter loops: range

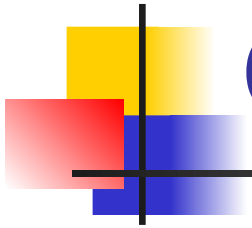
- The range function is really independent of for loops; although it’s used most often to generate indexes in a for loop
- There are three formats of range:

```
>>> range(5)
>>> range(2,5)
>>> range(0,10,2)
```



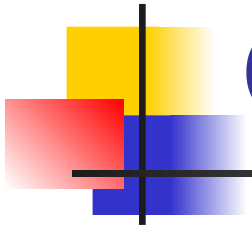
Counter loops: range

- With one argument, range generates a list with integers from zero up to but NOT including the argument’s value
- If you pass in two arguments, the first is taken as the lower bound
- An optional third argument can give a step; if used, Python adds the step to each successive integer in the result



Counter loops: range

```
>>> range(5)
[0, 1, 2, 3, 4]

>>> range(2,5)
[2, 3, 4]

>>> range(0,10,2)
[0, 2, 4, 6, 8]
```



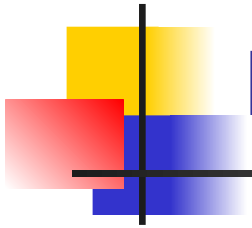
Counter loops: range

- Ranges can also be non-positive, and non-ascending, if you want them to be:

```
>>> range(-5,5)
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]

>>> range(5,-5,-1)
[5, 4, 3, 2, 1, 0, -1, -2, -3, -4]
```

range in for loop

```
>>> sum=0
>>> for i in range(100):
                sum = sum + i
>>> product=1
>>> for i in range(5,10)
                product = product * 10
```



range in for loop

```
>>> aa=[1,2,3,4]
>>> sum=0
>>> for i in aa:
                sum = sum + i


>>> sum=0
>>> for i in range(len(aa)):
>>>             sum = sum + aa[i]
```

range in for loop

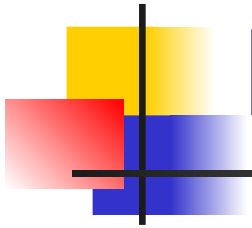
aa= [90,80,75,60,80,77,65,30,50,100]

- If we only want to compute the sum of first 5 scores:

```
>>> sum=0
>>> for i in range(5):
>>>         sum=sum + aa[i]
```

- If we only need the even number student's score:

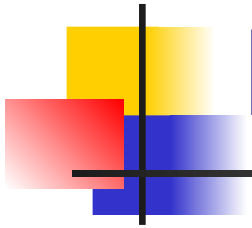
```
>>> sum=0
>>> for i in range(0,len(aa),2):
>>>         sum=sum + aa[i]
```



Finding Maximum

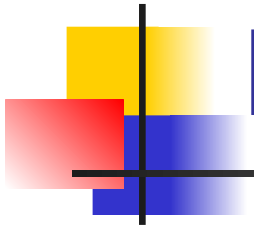
- Given a list, how to find the maximum value by using the for loop?

- Two approaches



Find Max

```
aa=[78, 95, 100, 60, 88, 93, 95, 80, 91]

max=0
for i in aa:
    if i > max:
        max=i

print max
```

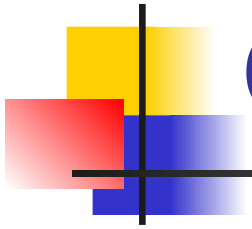


Find Max

```
aa=[60, 78, 95, 100, 60, 88, 93, 95, 80,
   91]


max=0
for i in range(len(aa)):
        if aa[i]>max:
            max=aa[i]
```

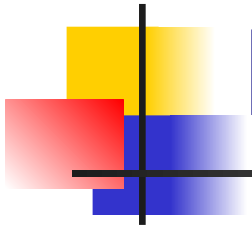
Find Max in even number index

```
aa=[60,78, 95, 100, 60, 88, 93, 95, 80, 91]

max=0
for i in range(0,len(aa),2):
        if aa[i]>max:
                max=aa[i]


print max
```



Class Practice

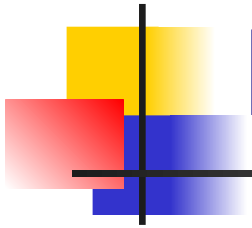
- Find minimum value in even number index



Pyramid

- Use for loop to print the Pyramid looks like: (User can input the height of the pyramid)

```
*
**
***
****
*****
******
*******
```



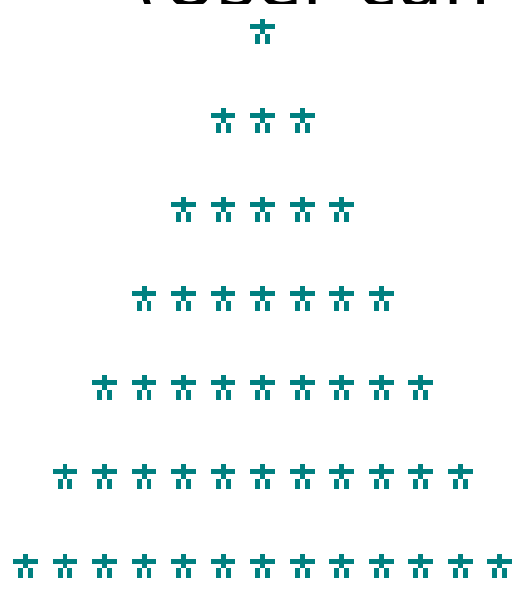
Pyramid

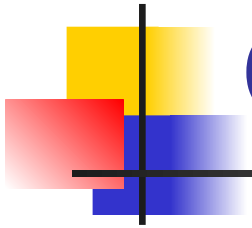
```
for i in range(8):
        print ‘*’ * i
```

Class Practice: Pyramid

- Use for loop to print the Pyramid looks like:

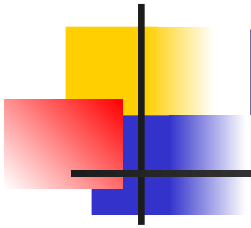
  (User can input the height of the





Overall Outline

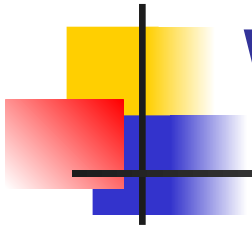
- If Statement
- Loop
- Function



Function

- In this chapter, we will move on to explore a set of additional statements that create functions of our own

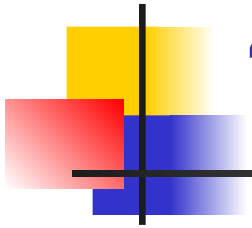
- In simple terms, a function is a device that **groups** a set of statements, so they can be run more than *once* in a program



Why Function?

- Functions serve two primary development roles:

1. **Code Reuse:** Functions allows us to group and generalize code to be used arbitrarily many times after it is defined.
2. **Procedure decomposition:** Functions also provide a tool for splitting systems into pieces---one function for each subtask.



“def” statement

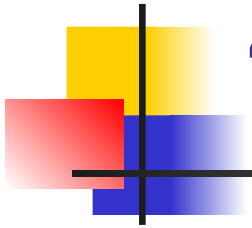
- The def statement creates a function object and assigns it to a name. the def general format:

```
def <name> ( ):
        <statements>
```

- the statement block becomes the function’s body---the code Python executes each time the function is called

Function Example

```
def star():
    num=int(input("please input a number: "))
    for i in range(num+1):
        print ('*' * i)



star()   # we call “star” function once
```



“def” statement

- Now we try to see the def format with arguments:

def <name>(arg1, arg2, ..., argN):

    <statement>

Function Example with one Argument

```
def star(num):
    for i in range(num+1):
        print ('*' * i)



for i in range(3,12,3):
    star(i)   # we call “star” function 3 times here
```

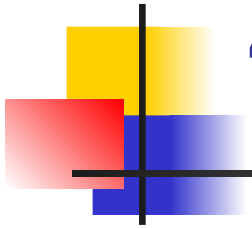
Function Example with 2 argument

```
def multiply(x,y):
   value=x*y
   print (value)



multiply(4,10)
multiply(3,6)
multiply(2,18)
```

Function Example with 3 argument

```
def multiply(x,y,z):
    value=x*y*z
    print value
```

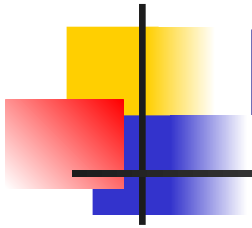
```
multiply(4,10,1)    #generate result 40
multiply(4,10)      #error message: multiply()
                       takes exactly 3 arguments
                    (2 given)
```



“def” statement

- Now we try to see the def format with arguments and **return** function:

```
def <name>(arg1, arg2, ..., argN):
        ...
        return <value>
```



Function Example

```
def times(x,y):
    value=x*y
    return value


aa = times(2,4)    #aa=8
aa = times(3.14,4)#aa=12.56
aa = times(“ha”,4)#aa=“hahahaha”
list=[1,2,3,4]
aa = times(list,2)  #aa=[1,2,3,4,1,2,3,4]
```

Class Practice

- Please write a function that take “one” variable (range from 1 to 13) and the function will print out a poker card figure.
- Such as: