Lecture 16

Topics to be covered:

Chapter 9:
- Nested loops
- Developing programs incrementally
- Break and continue
- Loop else
- Getting both index and value when looping: enumerate()
- Additional practice: Dice statistics

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
   print()
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 10

y → 1

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
    →print(x+y,end = " ")
   print()
```

x → 10

y → 1

```
11
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

y → 2

```
11 12
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 10

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
   print()
```

x → 20

```
11 12 13 14 15 16 17 18 19
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 20

y → 1

11 12 13 14 15 16 17 18 19

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
    print(x+y,end = " ")
    print()
```

x → 20

y → 1

```
11 12 13 14 15 16 17 18 19
21
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 20

y → 2

```
11 12 13 14 15 16 17 18 19
21
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
  ⟶ print(x+y,end = " ")
   print()
```

x → 20

y → 2

```
11 12 13 14 15 16 17 18 19
21 22
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 20

y → 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 30

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 30

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
    for y in range(1,10):
        print(x+y,end = " ")
    print()
```

x → 30

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
  print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
  print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

How many entries are in the table on the left?

Nested loops

A loop that appears as part of the body of another loop is called a nested loop.

The nested loops are commonly referred to as the outer loop and inner loop.

```
for x in [10,20,30,40]:
  for y in range(1,10):
     print(x+y,end = " ")
  print()
```

x → 10, 20, 30, 40

y → 1, 2, 3, 4, 5, 6, 7, 8, 9

```
11 12 13 14 15 16 17 18 19
21 22 23 24 25 26 27 28 29
31 32 33 34 35 36 37 38 39
41 42 43 44 45 46 47 48 49
```

How many entries are in the table on the left?

$4 \times 9 = 36$

Nested loops

Example: two-letter domain names

```
print('Two-letter domain names:')

letter1 = 'a'
letter2 = '?'

while letter1 <= 'z':  # Outer loop

    letter2 = 'a'

    while letter2 <= 'z':  # Inner loop
        print('%s%s.info' % (letter1, letter2))
        letter2 = chr(ord(letter2) + 1)
    letter1 = chr(ord(letter1) + 1)
```

Nested loops

Example: two-letter domain names

```
print('Two-letter domain names:')

letter1 = 'a'
letter2 = '?'

while letter1 <= 'z':   # outer loop
    letter2 = 'a'
    while letter2 <= 'z':   # inner loop
        print('%s%s.info' % (letter1, letter2))
        letter2 = chr(ord(letter2) + 1)
    letter1 = chr(ord(letter1) + 1)
```

```
Two-letter domain names:
aa.info
ab.info
ac.info
ad.info
ae.info
af.info
ag.info
ah.info
...
zx.info
zy.info
zz.info
```

Break and continue

A break statement in a loop causes an immediate exit from the loop.

```
while True:
   x = int(input("Enter an integer > 10:"))
   if x > 10:
      break
print("you made it!")
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 5
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 1
i = 5

5 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 5

5 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
   → continue

print("s =", s)
```

```
s = 1
i = 5

5 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 1
i = 6
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 1
i = 6

6 % 2 = 0
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

s = s*(6//2) = 1*3 = 3
i = 6

6 % 2 = 0

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 3
i = 7
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

s = 3
i = 7

7 % 2 = 1

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 3
i = 7

7 % 2 = 1
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 3
i = 8
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

```
s = 3
i = 8

8 % 2 = 0
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

    if i%2 == 0:
        s *= (i//2)

    else:
        continue

print("s =", s)
```

s = 3 * (8//2) = 12
i = 8

8 % 2 = 0

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s = 12
i = 9

.....
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s =  181440
i = 19
```

Break and continue

A continue statement in a loop causes an immediate jump to the while or for loop header statement.

```
s = 1

for i in range(5,20):

   if i%2 == 0:
      s *= (i//2)

   else:
      continue

print("s =", s)
```

```
s =  181440
i = 19
```

s = 181440

Loop else

A loop may include an else statement: it executes only if the loop terminates “normally” (not using a break statement).

Loop else

Example: I'm asking the user to input *5 positive integer values*. If the user enters all 5 numbers that are positive integers, I will display their sum , their product and their average. If a user enters 0 or a negative integer, I will display nothing.

```
counter = 0, s = 0, product = 1
for i in range(5):
   a = int(input("enter a positive integer:"))
   if a <= 0:
      break
   else:
      s += a
      product *= a
      counter += 1
else: print("Their sum is %d, their product is %d, and their average if %f" % (s, product, s/counter) )
```

Loop else

Example: I'm asking the user to input *5 positive integer values*. If the user enters all 5 numbers that are positive integers, I will display their sum , their product and their average. If a user enters 0 or a negative integer, I will display nothing.

```
counter = 0, s = 0, product = 1
for i in range(5):
   a = int(input("enter a positive integer:"))
   if a <= 0:
      break
   else:
      s += a
      product *= a
      counter += 1
else: print("Their sum is %d, their product is %d, and their average if %f" % (s, product, s/counter) )
```

open the program
5positiveIntegers.py
and run it

Getting both index and value when looping: enumerate()

Sometimes we need to run a loop that:

1. iterates over the elements in the container, and
2. at the same time we have access to the position of the element in the container.

Getting both index and value when looping: enumerate()

Sometimes we need to run a loop that:
1. iterates over the elements in the container, and
2. at the same time we have access to the position of the element in the container.

Check this code fragment out:

```
myList=["Mara","Lina","Alexis","Carol"]

for i,item in enumerate(myList):
  print(i,item)
```

```
0 Mara
1 Lina
2 Alexis
3 Carol
```

see the program enumerate1.py

Additional practice: Dice statistics

Let's *simulate* rolls of two dice:

The die faces are 1 …. 6

So when we through two dice, we can get a score of 2 … 12!

We roll the dice randomly! Python has random library that will help us to *simulate* a roll of two dice.

```
import random

random.randint(1,6) # returns a pseudo-random number
# between 1 and 6, including.
```

Additional practice: Dice statistics

Let's *simulate* rolls of two dice:

```
import random

print("first roll:", random.randint(1,6))
print("second roll:", random.randint(1,6))
```

```
first roll: 4
second roll: 6
```

Additional practice: Dice statistics

Let's *simulate* rolls of two dice:

```
import random

print("first roll:", random.randint(1,6))
print("second roll:", random.randint(1,6))
```

```
first roll: 4
second roll: 6
```

see the program twoDiceRolls.py

Additional practice: Dice statistics

Let's *simulate* 100 rolls of two dice and see the statistics: how many time of each value between 2 and 12 did we get!

see the program twoDice100Rolls.py