

Fundamentals of Programming

Unit 4

Control Statements, Numeric & Lists

Loop Control Statements

Use control statements with your looping!

Loop Control Statements

Loop Control Statements

change execution from its normal sequence

❑ Features of Looping

- when execution leaves a scope, all automatic objects that were created in that scope are destroyed

❑ Types of Control Statements

- *break* statement
- *continue* statement
- *pass* statement

break Statement

break Statement

terminate the current loop and resume execution at the next statement

❑ Features of *break* Statement

- *break* statement can be used in both while and for loops
- the most common use for break is when some external condition is triggered requiring a hasty exit from a loop
- using nested loops → the break statement stops the execution of the innermost loop and start executing the next line of code after the block

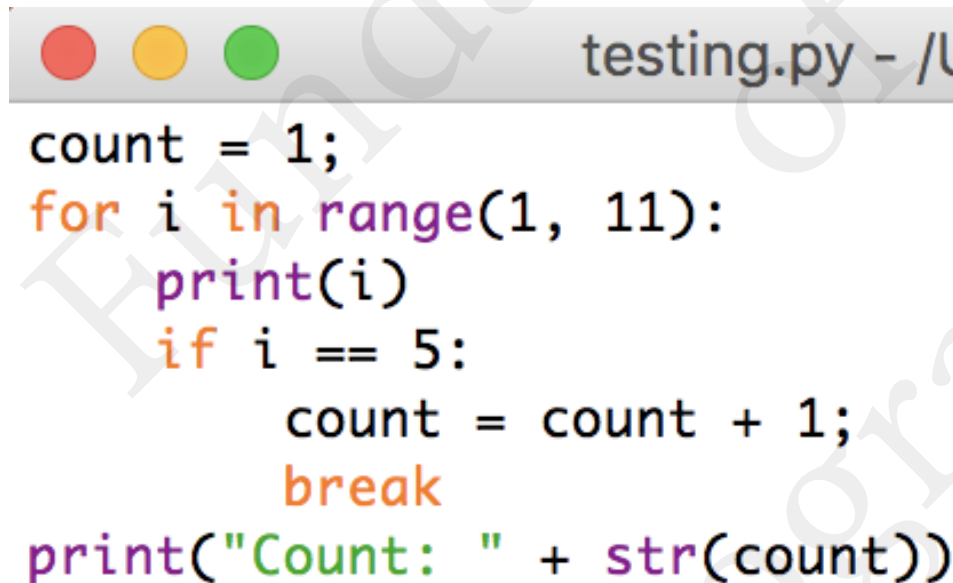
❑ Syntax

break

break Statement

□ Example

- Input the following code and see the results



```
count = 1;
for i in range(1, 11):
    print(i)
    if i == 5:
        count = count + 1;
        break
print("Count: " + str(count))
```

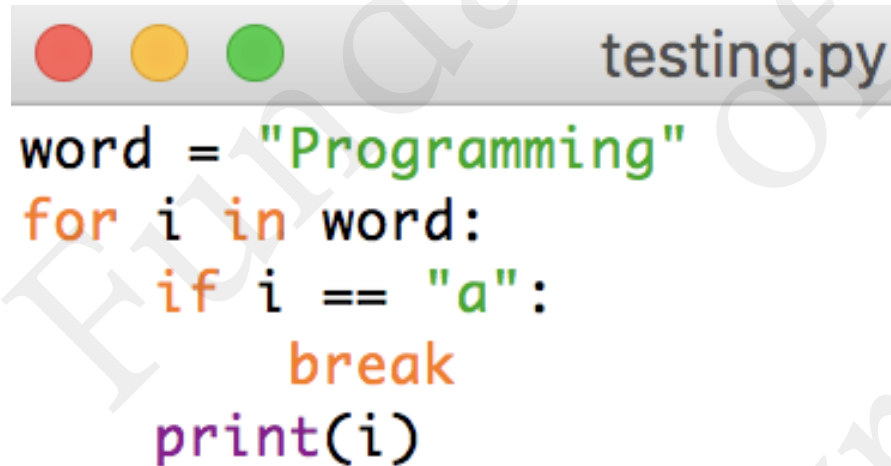
Programming Code

```
count = 1;
for i in range(1, 11):
    print(i)
    if i == 5:
        count = count + 1;
        break
print("Count: " + str(count))
```

break Statement

□ Example

- Input the following code and see the results



```
word = "Programming"
for i in word:
    if i == "a":
        break
print(i)
```

Programming Code

```
word = "Programming"
for i in word:
    if i == "a":
        break
    print(i)
```

break Statement

❑ Class Activity

- Ask the user to input a positive integer (assume that integer is smaller than or equal to 100). Display each integer from that integer to 100. The program will stop if it first meets the integer which can be divided by 4. Sample output is shown below.

```
Enter a positive integer: 97
```

```
97
```

```
98
```

```
99
```

```
100
```

```
The first integer meet that can divided by 4: 100
```

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

```
5
```

```
6
```

```
7
```

```
8
```

```
The first integer meet that can divided by 4: 8
```

continue Statement

continue Statement

reject all the remaining statements in the current iteration of the loop
and move the control back to the top of the loop

❑ Features of *continue* Statement

- *continue* statement can be used in both while and for loops

❑ Syntax

continue

continue Statement

□ Example

- Input the following code and see the results



```
i = 0
while(i < 5):
    i = i+1
    if(i == 3):
        continue
    print(i)
```


Programming Code

```
i = 0
while(i < 5):
    i = i+1
    if(i == 3):
        continue
    print(i)
```

continue Statement

□ Example

- Input the following code and see the results



```
word = "Program"  
for i in word:  
    if(i == "a"):  
        continue  
    print(i)
```

Programming Code

```
word = "Program"  
for i in word:  
    if(i == "a"):  
        continue  
    print(i)
```

continue Statement

❑ Class Activity

- Ask the user to input a sentence. Display the sentence but skip all the spaces in it. Sample output is shown below.

Enter a sentence: I am learning Python
IamlearningPython

pass Statement

pass Statement

used when a statement is required syntactically but you do not want any command or code to execute

❑ Features of *pass* Statement

- a null operation; nothing happens when it executes
- useful in places where the code will eventually go, but has not been written yet

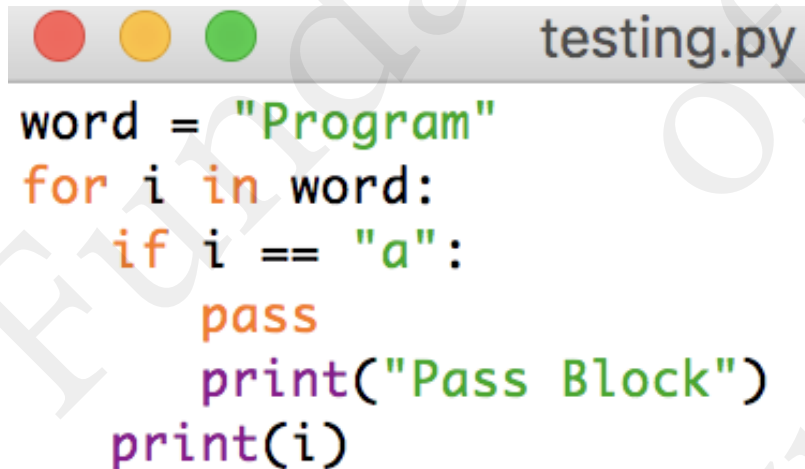
❑ Syntax

pass

pass Statement

□ Example

- Input the following code and see the results



```
word = "Program"
for i in word:
    if i == "a":
        pass
    print("Pass Block")
print(i)
```

Programming Code

```
word = "Program"
for i in word:
    if i == "a":
        pass
    print("Pass Block")
print(i)
```

More about Numbers

Learn more about numbers...

Numbers

□ Numbers

- number data types store numeric values
- immutable data types → changing the value of a number data type results in a newly allocated object

□ Types of Numerical Types

- int (signed integers) → positive or negative whole numbers with no decimal point
- long (long integers) → integers of unlimited size
- float (floating point real values) → real numbers and are written with a decimal point dividing the integer and fractional parts
- complex (complex numbers) → the form $a + bj$, real part of the number is a , and the imaginary part is b

Number Type Conversion

□ Number Type Conversion

- Python converts numbers internally in an expression containing mixed types to a common type for evaluation
- sometimes, we need to coerce a number explicitly from one type to another to satisfy the requirements of an operator or function parameter

Number Type Conversion

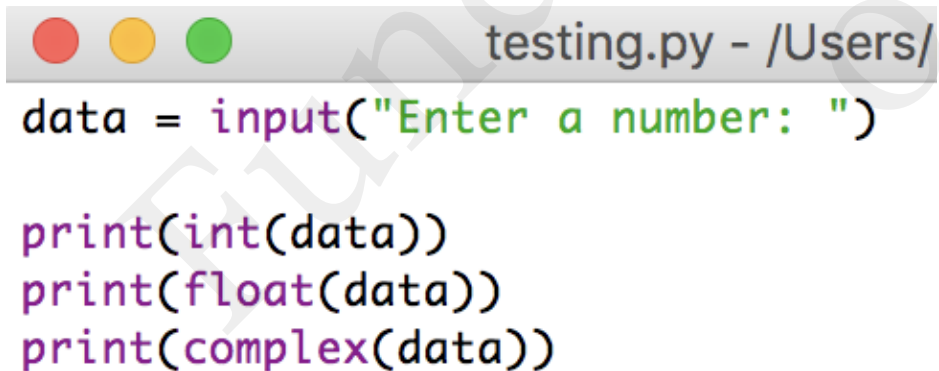
□ Kinds of Number Type Conversion

Function	Description
<code>int(x)</code>	convert x to a plain integer
<code>long(x)</code>	convert x to a long integer (not in Python 3.x)
<code>float(x)</code>	convert x to a floating-point number
<code>complex(x)</code>	convert x to a complex number with real part x and imaginary part zero
<code>complex(x, y)</code>	convert x and y to a complex number with real part x and imaginary part y x and y are numeric expressions

Number Type Conversion

□ Example

- Input the following code and see the results



A terminal window titled 'testing.py - /Users/' with three colored window control buttons (red, yellow, green) on the left. The code inside is as follows:

```
data = input("Enter a number: ")

print(int(data))
print(float(data))
print(complex(data))
```

Programming Code

```
data = input("Enter a number: ")

print(int(data))
print(float(data))
print(complex(data))
```

Mathematical Constants

□ Mathematical Constants

- π → The mathematical constant π
- e → The mathematical constant e

□ Example

- Input the following code and see the results



```
import math
```

```
print(math.pi)
```

```
print(math.e)
```

Programming Code

```
import math
```

```
print(math.pi)
```

```
print(math.e)
```

Mathematical Constants

❑ Class Activity

- Ask the user to input the radius and calculate the perimeter of the circle. Sample output is shown below.
- Note: perimeter of the circle = $2 \times \pi \times \text{radius}$

Enter the radius: 5.5

Perimeter of the circle: 34.55751918948772

choice() Function

❑ choice() Function

- return a random item from a list, tuple, or string

❑ Parameters

- a list, tuple, or string

❑ Return Value

- a random item

choice() Function

□ Example

- Input the following code and see the results

```
import random
```

```
number = [10, 22, 34, 57]
```

```
print("Random Choice: ", random.choice(number))
```

Programming Code

```
import random
```

```
number = [10, 22, 34, 57]
```

```
print("Random Choice: ", random.choice(number))
```

randrange() Function

❑ randrange() Function

- return a randomly selected element from range(start, stop, step)

❑ Parameters

- start (optional) → start point of the range (included in the range)
- stop (required) → stop point of the range (excluded from the range)
- step (optional) → steps to be added in a number to decide a random number

❑ Return Value

- a random item from the given range

randrange() Function

□ Example

- Input the following code and see the results

```
import random
```

```
#generate a random number from 0 to 10
```

```
print ("randrange(11) : ", random.randrange(11))
```

```
#generate a random number from 1 to 10
```

```
print ("randrange(1, 11) : ", random.randrange(1, 11))
```

```
#generate a random number from 1 to 10, skipping with 3
```

```
print ("randrange(1, 11, 3) : ", random.randrange(1, 11, 3))
```


randrange() Function

□ Example

- Input the following code and see the results

Programming Code

```
import random
```

```
#generate a random number from 0 to 10
```

```
print ("randrange(11) : ", random.randrange(11))
```

```
#generate a random number from 1 to 10
```

```
print ("randrange(1, 11) : ", random.randrange(1, 11))
```

```
#generate a random number from 1 to 10, skipping with 3
```

```
print ("randrange(1, 11, 3) : ", random.randrange(1, 11, 3))
```

shuffle() Function

❑ shuffle() Function

- randomize the items of a list in place

❑ Parameters

- a list or tuple

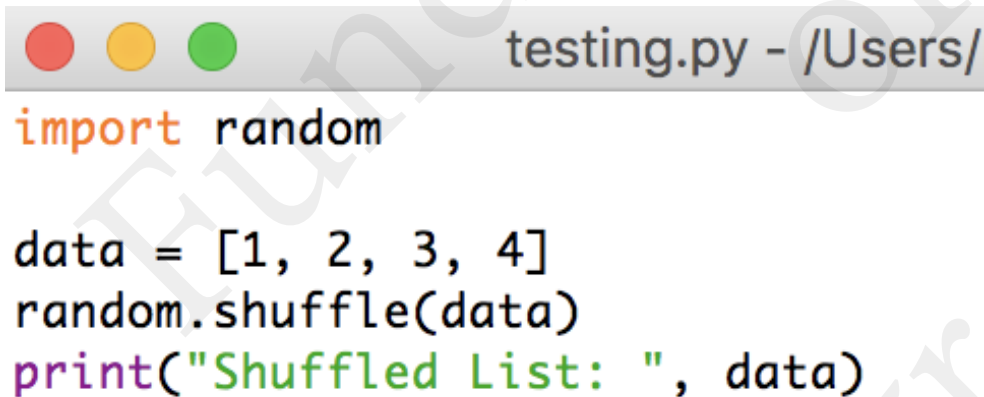
❑ Return Value

- do not return any value

shuffle() Function

□ Example

- Input the following code and see the results



```
import random

data = [1, 2, 3, 4]
random.shuffle(data)
print("Shuffled List: ", data)
```

Programming Code

```
import random

data = [1, 2, 3, 4]
random.shuffle(data)
print("Shuffled List: ", data)
```

More about Strings

Perform more operations with strings...

Escape Characters

❑ Escape Characters

- an escape character gets interpreted in a single quoted as well as double quoted strings
- a list of escape or non-printable characters that can be represented with backslash notation

❑ Common Types of Escape Characters

Escape Character	Description
<code>\n</code>	newline
<code>\t</code>	tab

String Special Operators

❑ Common String Special Operators

Operators	Description
in	Membership – return true if a character exists in the given string
not in	Membership - return true if a character does not exist in the given string
r/R	Raw String - suppress actual meaning of Escape characters

String Operations

□ Example

- Input the following code and see the results

```
data = "Python Programming"
if "Python" in data:
    print("\tWe have \nPython")
if "Java" not in data:
    print(r"\tWe do not have \nJava")
```

Programming Code

```
data = "Python Programming"
```

```
if "Python" in data:
    print("\tWe have \nPython")
```

```
if "Java" not in data:
    print(r"\tWe do not have \nJava")
```

String Formatting Operator

❑ String Formatting Operator

- the string format operator %
- unique to strings and makes up for the lack of having functions from C's printf() family

❑ Common Set of Symbols Using Along with %

Format Symbol	Conversion
%c	character
%s	string conversion via str() prior to formatting
%d	signed decimal integer
%f	floating point real number

String Operations

□ Example

- Input the following code and see the results

Programming Code

```
name = input("What is your name? ")  
age = int(input("How old are you? "))  
print("Hello, %s! You are %d years old." % (name, age))
```

```
name = input("What is your name? ")  
age = int(input("How old are you? "))  
  
print("Hello, %s! You are %d years old." % (name, age))
```

String Common Methods

❑ upper()

- return a copy of the string in which all case-based characters have been uppercased

❑ lower()

- return a copy of the string in which all case-based characters have been lowercased

❑ capitalize()

- return a copy of the string with only its first character capitalized

String Operations

□ Example

- Input the following code and see the results

Programming Code

```
sentence = input("Enter a sentence: ")
```

```
print("UPPER: ", sentence.upper())
```

```
print("lower: ", sentence.lower())
```

```
print("Capitalize: ", sentence.capitalize())
```

```
sentence = input("Enter a sentence: ")
```

```
print("UPPER: ", sentence.upper())
```

```
print("lower: ", sentence.lower())
```

```
print("Capitalize: ", sentence.capitalize())
```

String Common Methods

❑ len()

- return the length of the string

❑ isalpha()

- check whether the string consists of alphabetic characters only

❑ isnumeric()

- check whether the string consists of only numeric characters
- this method is present only on unicode objects

String Operations

□ Example

- Input the following code and see the results

Programming Code

```
sentence = input("Enter a sentence: ")  
  
print("Length: ", len(sentence))  
print("Is Alpha? ", sentence.isalpha())  
print("Is Numeric? ", sentence.isnumeric())
```

```
sentence = input("Enter a sentence: ")  
  
print("Length: ", len(sentence))  
print("Is Alpha? ", sentence.isalpha())  
print("Is Numeric? ", sentence.isnumeric())
```

String Operations

□ Class Activity

- Ask the user to input a sentence which can be with both upper and lower cases.
- Ask the user to further input the words that want to find
- Display the result as shown below using the string formatting operators

= RESTART:

```
Enter a sentence: Hello World  
Enter a word to be found: HELLo  
Found: HELLo
```

= RESTART:

```
Enter a sentence: hello WORLD  
Enter a word to be found: Python  
Python cannot be found
```

More about Lists

Perform more operations with lists...

Sequence

❑ Sequence

- the most basic data structure in Python
- each element of a sequence is assigned a number - its position or index
- the first index is zero, the second index is one, and so forth
- 6 built-in types of sequences, e.g. lists and tuples

❑ Operations with Sequence

- indexing, slicing, adding, multiplying, and checking for membership
- has built-in functions for finding the length of a sequence and for finding its largest and smallest elements

Lists

□ Lists

- a list of comma-separated values (items) between square brackets
- items in a list need not be of the same type
- list indices start at 0, and lists can be sliced, concatenated and so on

□ Create a List

- Review Example

```
listA = ['Python', 'Programming']  
listB = [123, 456, 789]
```

```
print(listA)  
print(listB)
```

Programming Code

```
listA = ['Python', 'Programming']  
listB = [123, 456, 789]
```

```
print(listA)  
print(listB)
```

Lists

❑ Access Values in List

- use the square brackets for slicing along with the index or indices to obtain value available at that index

❑ Example

```
listA = ['Python', 'Programming', 'Learning', 'Happy']  
print("First item: ", listA[0])  
print("2nd to 4th items: ", listA[1:4])
```

Programming Code

```
listA = ['Python', 'Programming', 'Learning', 'Happy']  
print("First item: ", listA[0])  
print("2nd to 4th items: ", listA[1:4])
```

Lists

❑ Add Values into List

- add to elements in a list with the ***append()*** method

❑ Example

```
listA = ['Python', 'Programming', 'Learning', 'Happy']
```

```
listA.append('School')
```

```
print(listA)
```

Programming Code

```
listA = ['Python', 'Programming', 'Learning', 'Happy']
```

```
listA.append('School')
```

```
print(listA)
```

Lists

❑ Update Values in List

- update the original value in the list
- update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator

❑ Example `listA = ['Python', 'Programming', 'Learning', 'Happy']`

```
print("Original second item: ", listA[1])  
listA[1] = 'Program'  
print("Updated second item: ", listA[1])
```

Programming Code

```
listA = ['Python', 'Programming', 'Learning', 'Happy']  
print("Original second item: ", listA[1])  
listA[1] = 'Program'  
print("Updated second item: ", listA[1])
```

Lists

❑ Delete Values in List

- use either the ***del*** statement if we know exactly which element(s) you are deleting or the ***remove()*** method if we do not know

❑ Example

Programming Code

```
listA = ['Python', 'Programming',  
'Learning', 'Happy']
```

```
print(listA)
```

```
del listA[2]
```

```
print(listA)
```

```
listA.remove('Python')
```

```
print(listA)
```

```
listA = ['Python', 'Programming', 'Learning', 'Happy']
```

```
print(listA)
```

```
del listA[2]
```

```
print(listA)
```

```
listA.remove('Python')
```

```
print(listA)
```

List Common Methods

❑ len()

- return the number of elements in the list

❑ sort()

- sort objects of list

❑ count()

- return count of how many times object occurs in list

List Operations

□ Example

- Input the following code and see the results

Programming Code

```
myList = [1, 3, 2, 4, 7, 3, 1, 1]
print("Length: ", len(myList))
myList.sort()
print("Sorted List: ", myList)
print("Number of 1: ", myList.count(1))
```

```
myList = [1, 3, 2, 4, 7, 3, 1, 1]
print("Length: ", len(myList))
myList.sort()
print("Sorted List: ", myList)
print("Number of 1: ", myList.count(1))
```

List Operations

❑ Class Activity

- Ask the user to input the grades of students (from A to E) until user input 'Z' to end the input
- Ask the user to input one of the grade (from A to E) and help him/her to check the percentage of that grade within those grades input
- Display the results as shown below

= RESTART:

```
Enter the grade(A - E, Z to end): a
Enter the grade(A - E, Z to end): B
Enter the grade(A - E, Z to end): A
Enter the grade(A - E, Z to end): C
Enter the grade(A - E, Z to end): c
Enter the grade(A - E, Z to end): d
Enter the grade(A - E, Z to end): E
Enter the grade(A - E, Z to end): a
Enter the grade(A - E, Z to end): z
The grade list is:
['A', 'B', 'A', 'C', 'C', 'D', 'E', 'A']
Enter the grade that you want to check: a
Percentage of Grade A = 37.5 %
```


Hope you enjoy the class

Thank you