

CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding Sep 9 2019, Lecture 6

Admins

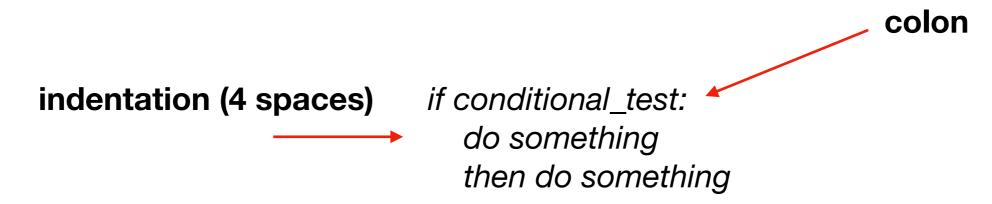
- Office Hours: Tu 3:00pm-4:00pm, Caruth Hall, 4th floor, adjunct faculty room.
- Help Desk Schedule, Mon-Fri 8:30am-5:00pm, walk in, Caruth Hall 484
- Lab sessions start this week, first lab posted

Agenda

- Agenda:
 - Quick review of concepts from last lecture
 - while/for loops
 - Functions and Modules

if statements

- if statements
 - if the conditional test is True, then execute the following statements, otherwise ignore.
- if statements syntax



if statements

- if-elif-else chain
 - When you need to test more than two possible situations.

```
age = 12
 3
        if age < 4:
            price = 0
        elif age < 18:</pre>
            price = 5
 7
        else:
            price = 10
 8
 9
        print("Your admission cost is $" + str(price))
10
        elif age < 18
if_statements ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/if_statements.py
Your admission cost is $5
```

if statements

Use if-elif chain vs Use multiple simple if statements

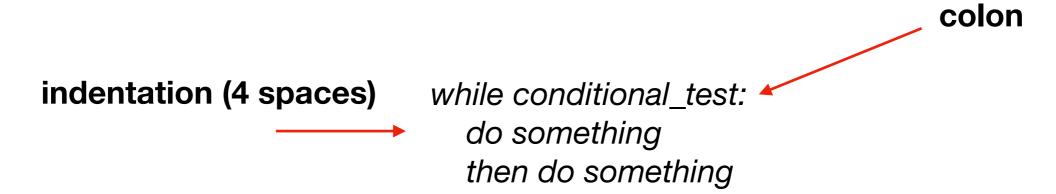
```
requested_toppings = ["mushrooms", "extra cheese"]
        if "mushrooms" in requested_toppings:
 4
            print("Adding mushrooms.")
 5
        elif "pepperoni" in requested_toppings:
            print("Adding pepperoni.")
 9
        elif "extra cheese" in requested_toppings:
10
            print("Adding extra cheese.")
11
12
        print("Finished making your pizza!")
13
         elif "pepperoni" in requested_t...
if statements >
 /Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/if_statements.py
 Adding mushrooms.
 Finished making your pizza!
```

```
requested_toppings = ["mushrooms", "extra cheese"]
 2
 3
4
        if "mushrooms" in requested_toppings:
            print("Adding mushrooms.")
 5
 6
        if "pepperoni" in requested_toppings:
            print("Adding pepperoni.")
        if "extra cheese" in requested_toppings:
10
            print("Adding extra cheese.")
11
        print("Finished making your pizza!")
12
13
if_statements
 /Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/if_statements.py
 Adding mushrooms.
 Adding extra cheese.
 Finished making your pizza!
```

Python loops

- if statements allow you to execute different piece of code based on the different situations (conditional test)
- Loops allow you to execute the same piece of code multiple times
- Python has two primitive loop commands
 - while loops
 - for loops

while loop syntax



- It will keep execute the code block as long as the conditional test is true.
 - usually you will need to modify the the values used in the conditional test once some conditions are met

A simple example

```
current_number = 1
while current_number)

while loops x
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py

1
2
3
4
5
```

The += operator is shorthand for current_number = current_number + 1

- Using break to exit a loop
 - To exit a loop immediately without running any remaining code in the loop

```
current_number = 1
while current_number <= 5:
    print(current_number)
    current_number += 1
    if current_number > 3:
    break

while current_number <= 5 > if current_number > 3

while_loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py
1
2
3
Process finished with exit code 0
```

- Using continue in a loop
 - Rather than breaking out of a loop entirely without executing the rest of its code, you can use the continue statement to return to the beginning of the loop based on the result of a conditional test

```
current_number = 0
current_number < 10:
current_number += 1
if current_number % 2 == 0:
continue
print(current_number)

itches and Consoles t_number < 10 > if current_number % 2 == 0
while_loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py
1
3
5
7
9
Process finished with exit code 0
```

Avoid infinite loops

```
while x = 1
print(x)
while_loops x
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py
1
2
3
4
5
Process finished with exit code 0
```

If you omit this, the loop will run forever



Use while loop with lists and dictionaries, example 1

```
unconfirmed_users = ["alice", "brain", "candace"]
        confirmed users = []
 2
 3
 4
       # Verify each user until there are no more unconfirmed users
       △# Move each verified user into the list of confirmed users.
 6
 7
        while unconfirmed_users:
            current user = unconfirmed users.pop()
 8
            print("Verifying user:" + current_user)
 9
10
            confirmed_users.append(current_user)
11
12
        # Display all confirmed users.
13
        print(confirmed_users)
while_loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py
Verifying user:candace
Verifying user:brain
Verifying user:alice
 ['candace', 'brain', 'alice']
Process finished with exit code 0
```

Use while loop with lists and dictionaries, example 2

```
pets = ["dog", "cat", "goldfish", "cat", "dog", "snake", "rabbit"]

print(pets)

while "cat" in pets:
    pets.remove("cat")

print(pets)

while_loops ×

/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py
['dog', 'cat', 'goldfish', 'cat', 'dog', 'snake', 'rabbit']
['dog', 'goldfish', 'dog', 'snake', 'rabbit']
Process finished with exit code 0
```

Use while loop with lists and dictionaries, example 3

```
employee_info = {
            "123": {
 2
                "name":"Joe",
 3
                "department": "CS"
 5
            "456": {
 6
                "name": "David",
                "department": "Math"
 8
 9
            "789": {
10
11
                "name": "Carl",
                "department": "CS"
12
13
14
15
16
        retired_ids = ["456", "789"]
17
18
        while retired ids:
            r_id = retired_ids.pop()
19
20
            del employee_info[r_id]
21
22
        print(employee_info)
23
         while retired_ids
while_loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/while_loops.py
{'123': {'name': 'Joe', 'department': 'CS'}}
Process finished with exit code 0
```

for loop

- For-each is Python's only form of for loop, this is less like the for keyword in other programming languages.
- A for loop steps through each of the items in a collection type (list, dictionary, etc) or any other type of object which is "iterable" (remember when we call .keys() method of a dictionary)
- Often used with lists and dictionaries

```
indentation (4 spaces)

for <each item> in <collection>:

<statements>
```

for loop

 if <collection> is a list or a tuple, then the loop steps through each element of the sequence

```
fruits = ["apple", "banana", "cherry"]
for f in fruits:
    print(f)

loops ×

/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py
apple
banana
cherry
```

 if <collection> is a string, then the loop steps through each character of this string

```
fruits = "apple"
for f in fruits:
    print(f)

loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py
a
p
p
l
e
```

for loop

Calculate the sum of a list

```
1    a_list = [3, 4, 52, 1, 3, 45, 100, 12]
2    total_sum = 0
3    for number in a_list:
4         total_sum += number
5    print(total_sum)

loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py
220
```

or use built in function

```
1    a_list = [3, 4, 52, 1, 3, 45, 100, 12]
2    total_sum = sum(a_list)
3
4    print(total_sum)
5

loops x
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py
220
```

Making numerical lists

- Many reasons exist to store a set of numbers
 - keep track of the positions of each character in a game
 - keep track of a player's scores
 - store temperatures for data visualization
 - •

 Lists are ideal for storing sets of numbers, and Python provides a number of tools to help you work efficiently with lists of numbers.

Using the range() Function

 Python's range() function makes it easy to generate a series of numbers. For example

```
for value in range(1, 5):
    print(value)

for value in range(1, 5)

loops ×

/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py
1
2
3
4
```

Note: range() here gives you 1 through 4, not 5. This behavior is called off-by-one. We have seen this when we used slicing to return a subset of a list

Using range() to Make a List of Numbers

 Call range() does not give you a list, wrap list() around a call to the range() function to get a list.

```
numbers = range(1, 5)
print(numbers)

numbers_list = list(range(1, 5))
print(numbers_list)

even_numbers = list(range(2, 11, 2))
print(even_numbers)

loops ×

/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py
range(1, 5)
[1, 2, 3, 4]
[2, 4, 6, 8, 10]
```

Using the enumerate() Function

Use enumerate() function to get the index and elements.

```
names = ['alice', 'bob', 'carl']
ages = [18, 32, 22]

for i, item in enumerate(names):
    print(i)
    print(item)

for i, item in enumerate(names)

loops ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/loops.py

alice
1
bob
2
carl
```

Demo

