CSC110 Lecture 7: If Statements

David Liu, Department of Computer Science

Navigation tip for web slides: press ***?*** *to see keyboard navigation controls.*

Announcements & Today's Plan

- Assignment 1 has been posted!
  - Check the FAQ (+ corrections) page
  - Additional TA office hours (schedule on Quercus)
  - **(NEW)** Academic Integrity in CSC110 advice page
- Join a Recognized Study Group
- Preps:
  - Grades and general feedback for Prep 2 will be released today
  - Prep 3 will also be released today

Last time on CSC110

> *Mathematical logic is a* language of boolean expressions.

Last class, we learned how to understand and translate statements written in symbolic logic.

We then used booleans expressions to write filtering comprehensions to extract elements of a collection:

```
>>> numbers = [1, 2, 3, 4]
>>> [x * 100 for x in numbers if x % 2 == 0]
[200, 400]
```

Today, we'll use boolean expressions to perform conditional execution of Python statements.

Today you'll learn to...

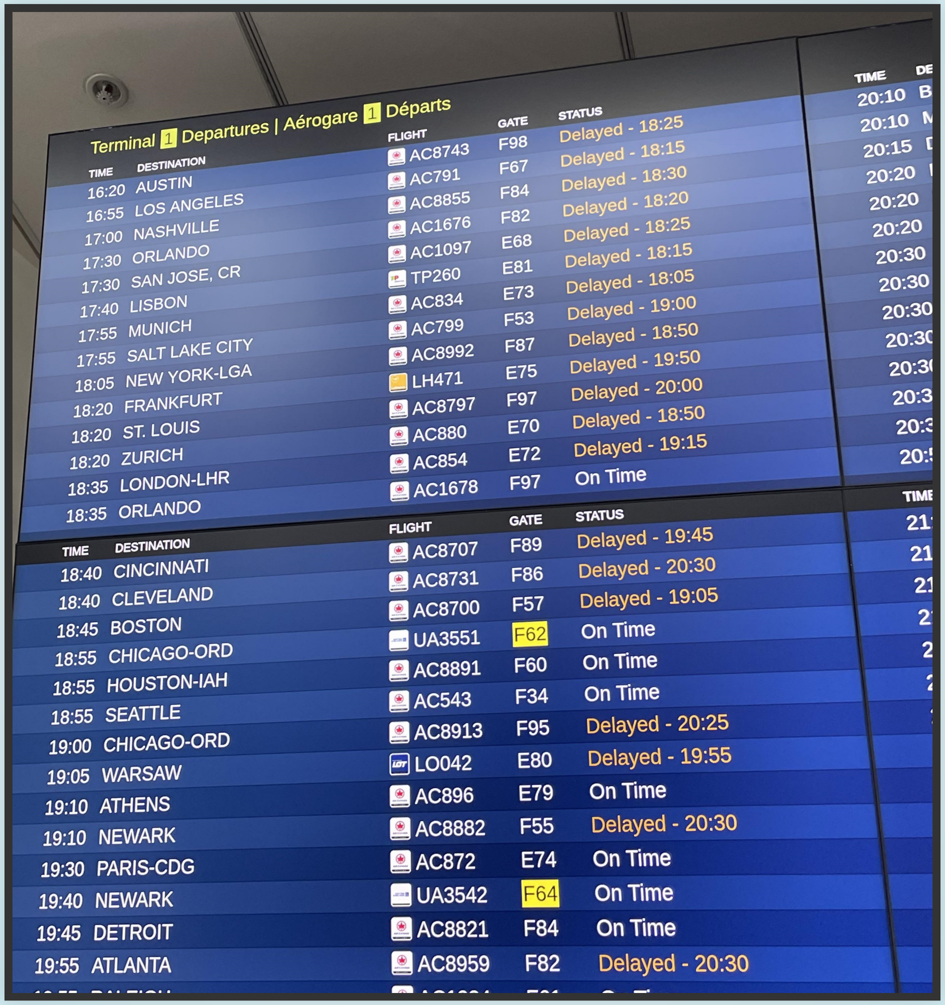
1. Use if statements to perform conditional execution of Python code.
2. Choose cases for if statements based on a problem description.
3. Simplify complex if statements.
4. Run PythonTA to check your code on preps and assignments.

Conditional execution

So far, all of our code has consisted of a sequence of statements executed one after the other.

```
def calculate_distance(x1: float, y1: float,
                       x2: float, y2: float) -> float:
    """Return the distance between points (x1, y1)
    and (x2, y2), rounded to 1 decimal place.
    """
    dx_squared = (x2 - x1) ** 2
    dy_squared = (y2 - y1) ** 2
    exact_distance = (dx_squared + dy_squared) ** 0.5
    return round(exact_distance, 1)
```

Flight statuses!


Terminal 1 Departures | Aérogare 1 Départs

TIME	DESTINATION	FLIGHT	GATE	STATUS
16:20	AUSTIN	AC8743	F98	Delayed - 18:25
16:55	LOS ANGELES	AC791	F67	Delayed - 18:15
17:00	NASHVILLE	AC8855	F84	Delayed - 18:30
17:30	ORLANDO	AC1676	F82	Delayed - 18:20
17:30	SAN JOSE, CR	AC1097	E68	Delayed - 18:25
17:40	LISBON	TP260	E81	Delayed - 18:15
17:55	MUNICH	AC834	E73	Delayed - 18:05
17:55	SALT LAKE CITY	AC799	F53	Delayed - 19:00
18:05	NEW YORK-LGA	AC8992	F87	Delayed - 18:50
18:20	FRANKFURT	LH471	E75	Delayed - 19:50
18:20	ST. LOUIS	AC8797	F97	Delayed - 20:00
18:20	ZURICH	AC880	E70	Delayed - 18:50
18:35	LONDON-LHR	AC854	E72	Delayed - 19:15
18:35	ORLANDO	AC1678	F97	On Time

TIME	DESTINATION	FLIGHT	GATE	STATUS
18:40	CINCINNATI	AC8707	F89	Delayed - 19:45
18:40	CLEVELAND	AC8731	F86	Delayed - 20:30
18:45	BOSTON	AC8700	F57	Delayed - 19:05
18:55	CHICAGO-ORD	UA3551	F62	On Time
18:55	HOUSTON-IAH	AC8891	F60	On Time
18:55	SEATTLE	AC543	F34	On Time
19:00	CHICAGO-ORD	AC8913	F95	Delayed - 20:25
19:05	WARSAW	LO042	E80	Delayed - 19:55
19:10	ATHENS	AC896	E79	On Time
19:10	NEWARK	AC8882	F55	Delayed - 20:30
19:30	PARIS-CDG	AC872	E74	On Time
19:40	NEWARK	UA3542	F64	On Time
19:45	DETROIT	AC8821	F84	On Time
19:55	ATLANTA	AC8959	F82	Delayed - 20:30

TIME
20:10
20:10
20:15
20:20
20:20
20:20
20:30
20:30
20:30
20:30

```python
def get_status(scheduled: int, estimated: int) -> str:
    """Return the flight status for the given scheduled and estimated departure times.

    The times are given as integers between 0 and 23 inclusive, representing
    the hour of the day.

    The status is either 'On time' or 'Delayed'.

    >>> get_status(10, 10)
    'On time'
    >>> get_status(10, 12)
    'Delayed'
    """
```

```
>>> get_status(10, 10)
'On time'
>>> get_status(10, 12)
'Delayed'
```

In some cases, we want to return `'On time'`, and in some other cases, we want to return `'Delayed'`.

```python
def get_status(scheduled: int, estimated: int) -> str:
    """..."""

    return 'On time'   # ???

    return 'Delayed'   # ???
```

We need a way to execute a Python statement only some of the time, based on a condition (boolean expression).

If statements

```
if <condition>:
    <statement>
    ...
else:
    <statement>
    ...
```

To execute an if statement:

1. First, evaluate the `<condition>` expression.
2. If the condition's value is `True`, execute the statement(s) under the `if` (called the if branch).

   If the condition's value is `False`, execute the statement(s) under the `else` (called the else branch).

Terminology note

If statements are Python statements, but not just a single line of code.

Technically, if statements are a type of **compound statement**, meaning they can contain other Python statements nested within them.

To PyCharm!

```
get_status(10, 10)
```

```
def get_status(scheduled: int, estimated: int) -> str:
    """..."""
    if estimated <= scheduled:  # <-- This line gets execu
        return 'On time'        # <-- This line gets execu
    else:
        return 'Delayed'
```

```
get_status(10, 12)
```

```
def get_status(scheduled: int, estimated: int) -> str:
    """..."""
    if estimated <= scheduled:  # <-- This line gets execu
        return 'On time'
    else:
        return 'Delayed'        # <-- This line gets execu
```

Exercise 1: Practice with if statements

Multiple branches

Now suppose flights have three statuses:

- **On time**: estimated time is before or at scheduled time
- **Delayed**: estimated time is late, but by less than 4 hours
- **Cancelled**: estimated time is late by 4 or more hours

Sometimes, we want to divide our statements into more than two different branches.

```
if <condition1>:
    <statement>
    ...
elif <condition2>:
    <statement>
    ...
... # [any number of elif conditions and branches]
else:
    <statement>
    ...
```

To execute an if statement with `elif`s:

1. Evaluate the if/elif conditions one at a time, in top-down order.
2. Stop at the first condition that evaluates to `True`, and execute the statements under that condition (the **branch** of that condition).
3. If the conditions all evaluate to `False`, execute the else branch.

Exercise 2: If statements with multiple branches

Explaining `if __name__ == '__main__'`

Recall doctest

```
if __name__ == '__main__':
    import doctest
    doctest.testmod(verbose=True)
```

What is __name__?

`__name__` is a special variable set by the Python interpreter for every module.

When the module is imported by another file, `__name__` is the name of the module:

```
>>> import doctest
>>> doctest.__name__
'doctest'
```

What is `__name__`?

When the module is **run**, `__name__` is set to `'__main__'`.

`__name__ == '__main__'` evaluates to…

- `True` when module is being run
- `False` when module is being imported

Putting it together

```
if __name__ == '__main__':
    ...
```

This if statement is called the **main block** of a Python module.

- If the module is being run, the main block gets executed.
- If the modules is being imported, the main block doesn’t get executed.

Simplifying if statements

As we introduce more kind of Python statements, our code gets more and more complex.

If statements aren’t always necessary (1)

```
def is_even(n: int) -> bool:
    """Return wheter n is even."""
    if n % 2 == 0:
        return True
    else:
        return False
```

```
def is_even(n: int) -> bool:
    """Return wheter n is even."""
    return n % 2 == 0
```

If statements aren't always necessary (2)

```python
def cap_at_100(grade: float) -> float:
    """Return grade, or 100.0 if grade exceeds 100.0."""
    if grade > 100.0:
        return 100.0
    else:
        return grade
```

```python
def cap_at_100(grade: float) -> float:
    """Return grade, or 100.0 if grade exceeds 100.0."""
    return min(grade, 100.0)
```

Exercise 3: Simplifying if statements

PythonTA demo

What is PythonTA?

PythonTA is a program that performs checks Python programs for common issues in correctness, code design, and code style. (PythonTA does other things too, which we'll explore later in CSC110!)

Fun fact: PythonTA is developed by David and various students under his supervision!

Running PythonTA

On all current and future course work, the starter files we provide will contain some code in the main block for running PythonTA on that file.

```
if __name__ == '__main__':
    import python_ta
    python_ta.check_all(config={...})
```

Make sure to **fix all issues reported by PythonTA before submitting your work**!

Demo!

Tip: in PyCharm, use “Code -> Reformat Code” to fix common style issues (but fix other issues manually).

Summary

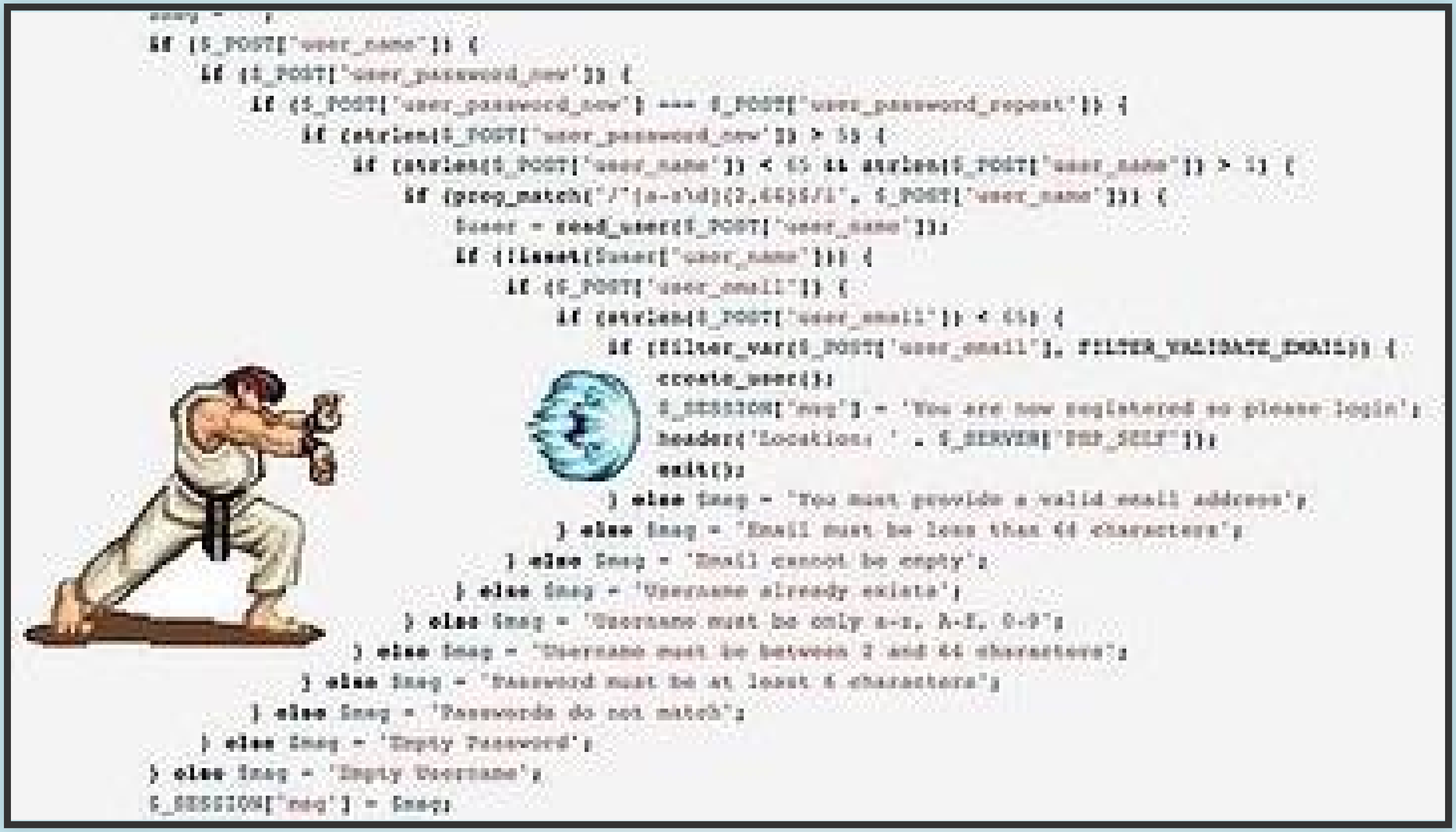
Today you learned to...

1. Use if statements to perform conditional execution of Python code.
2. Choose cases for if statements based on a problem description.
3. Simplify complex if statements.
4. Run PythonTA to check your code on preps and assignments.

Homework

- Readings:
  - Today's lecture: 3.4, 3.5, 3.6
  - Prep readings: 4.1, 4.2
  - For Monday: 4.1, 4.2, 4.3, 4.4
- Extra practice: try redoing today's flight examples using the `datetime.time` data type to represent times (see Section 2.5)
- Continue working on Assignment 1
- Prep 3 to be posted today

Term of the Day: Pyramid of Doom



```
if ($_POST['user_name']) {
    if ($_POST['user_password_new']) {
        if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
            if (strlen($_POST['user_password_new']) > 5) {
                if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                    if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                        $user = read_user($_POST['user_name']);
                        if (!isset($user['user_name'])) {
                            if ($_POST['user_email']) {
                                if (strlen($_POST['user_email']) < 65) {
                                    if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                        create_user();
                                        $_SESSION['msg'] = 'You are now registered so please login';
                                        header('Location: ' . $_SERVER['PHP_SELF']);
                                        exit();
                                    } else $msg = 'You must provide a valid email address';
                                } else $msg = 'Email must be less than 64 characters';
                            } else $msg = 'Email cannot be empty';
                        } else $msg = 'Username already exists';
                    } else $msg = 'Username must be only a-z, A-Z, 0-9';
                } else $msg = 'Username must be between 2 and 64 characters';
            } else $msg = 'Password must be at least 6 characters';
        } else $msg = 'Passwords do not match';
    } else $msg = 'Empty Password';
} else $msg = 'Empty Username';
$_SESSION['msg'] = $msg;
```

https://en.wikipedia.org/wiki/Pyramid_of_doom_(programming)