



THE UNIVERSITY OF
CHICAGO

MASTERS IN
COMPUTATIONAL
SOCIAL SCIENCE
THE UNIVERSITY OF CHICAGO

MACS 30111

Control Flow

Agenda

- **Misc**

- Who tested their code for str?
 - `str(867-5309)` vs `str("867-5309")` vs `str = "867-5309"`
- Is there any example of when "not" to use None as the default value?

- **HW00 grading**

- **SE1 due date:** TUESDAY (try to submit by Friday 9/29)
- Note: recommended reading from online version of book (not PDF – content is the same but numbering is a bit different)

AI AND YOU/ME/WE/US

- What does it mean for something to be your own work?
- What does it mean to use AI?
- What is / should our policy be?

Topics:

- ❑ Introduction
- ❑ *if else* conditional Statements
- ❑ *for* loops (sequence-based loops)
- ❑ *while* loops (condition-based loops)

Statements

Simple statements: assignments and the print function

```
n = 7  
print("n is", n)  
n = n + 10  
print("n is now", n)
```

These are four statements, which Python will execute sequentially.

A program is a sequence of *statements* that are run in the order in which they appear.

Control Flow

Sometimes, instead of running statements sequentially, we may want to alter the *control flow* of the program. For example:

- *I may only want to run some statements if a given condition is met: “Add a tax to the price, unless the customer is tax-exempt”*
- *I may want to run the some statements multiple times: “For every item in our inventory, increase the price by 5%”*

Imperative programming languages (e.g., Python) provide *conditional statements* and *looping statements* precisely to implement behaviors like these.

Topics:

- ❑ Introduction
- ❑ ***if else* conditional Statements**
- ❑ *for* loops (sequence-based loops)
- ❑ *while* loops (condition-based loops)

Conditional statements

For example:

```
if (n % 2) == 1:  
    print(n, "is odd")  
else:  
    print(n, "is even")
```

The basic structure:

```
if <boolean expression>:  
    <statements to run if True>  
else:  
    <statements to run if False>
```

When describing Python syntax, we will use <...> to denote placeholders.

A conditional statement allows the program to perform different actions based on the value of a boolean expression.

Focus on formatting

- Notice we don't NEED to use parentheses for the entire statement
 - But can, to make it easier for us to see / parse

For example:

```
if (n % 2) == 1:
    print(n, "is odd")
else:
    print(n, "is even")
```

The basic structure:

```
if <boolean expression>:
    <statements to run if True>
else:
    <statements to run if False>
```

Conditional statements

For example:

```
if n < 0:  
    print(n, "is negative")  
elif n % 2 == 1:  
    print(n, "is positive and odd")  
else:  
    print(n, "is positive and even")
```

Conditionals can also have multiple branches:

```
if <boolean expression>:  
    <block>  
elif <boolean expression>:  
    <block>  
else:  
    <block>
```

Quiz

Conditional statements involve using the following keywords ...

- if, otherwise
- if, else, default
- if, elif, else

A conditional statement in Python decides what branch to run by evaluating ...

- An arithmetic expression
- A boolean expression
- An expression that returns either one or zero

Topics:

- ❑ Introduction
- ❑ *if else* conditional Statements
- ❑ ***for* loops (sequence-based loops)**
- ❑ *while* loops (condition-based loops)

Loops

Loops provide a mechanism for **repeating** work in a program.

For example:

- Given a set of values, we may want to perform the same action on each of them.
- We may want to keep performing a certain action until a condition is true.

There are two types of loops: “*for*” loops and “*while*” loops.

“for” loops

Structure:

```
for <variable> in <sequence>:  
    <statements to run>
```

For example:

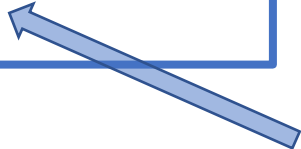
```
for n in [1, 4, 8, 9, 11]:  
    print(n)
```

Perform the same action on each of them in sequence

“for” loops

Structure:

```
for <variable> in <sequence>:  
    <statements to run>
```



```
for n in [1, 4, 8, 9, 11]:  
    if (n % 2) == 1:  
        print(n, "is odd")  
    else:  
        print(n, "is even")
```

“body” of the loop:

- Can contain multiple statements
- Conditional, loops

“for” loops

Using for loops to do something with all the integers in a given range.

```
for n in [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]:  
    if (n % 2) == 1:  
        print(n, "is odd")  
    else:  
        print(n, "is even")
```

The built-in range function:

```
for n in range(1, 21):  
    if (n % 2) == 1:  
        print(n, "is odd")  
    else:  
        print(n, "is even")
```


Primality Testing

Given an integer, determine whether it is prime or not:

- greater than 1
- can only be divided by itself and 1

Hint: combine loops and conditionals

Code

Work through: sketch

- Given an integer, determine whether it is prime or not:
 - Is the integer larger than 1?
 - Is the number divisible by anything smaller than itself?
- Code attempt:

```
var = 5
```

```
if var > 1:
```

```
    for num in range(2, var):
```

```
        if var % num == 0:
```

```
            print("var is not prime")
```

```
        else:
```

```
            print("var is prime")
```

```
else:
```

```
    print("integer is not prime")
```

Work through: sketch

- Given an integer, determine whether it is prime or not:
 - Is the integer larger than 1?
 - Is the number divisible by anything smaller than itself?

- Code attempt:

```
encountered_divisor = False
```

```
var = 83
```

```
for i in range(2, var):
```

```
    if var % i == 0:
```

```
        encountered_divisor = True
```

```
if encountered_divisor:
```

```
    print(var, "is NOT prime")
```

```
else:
```

```
    print(var, "is prime")
```

Quiz

What statement do we use to stop the execution of a loop?

- stop
- exit
- break
- endloop

Further example:

- What will this code do?

```
encountered_divisor = False
var = 80

for i in range(2, var):
    if var % i == 0:
        encountered_divisor = True
        print(i)
        break
if encountered_divisor:
    print(var, "is NOT prime")
else:
    print(var, "is prime")
```

Topics:

- ❑ Introduction
- ❑ *if else* conditional Statements
- ❑ *for* loops (sequence-based loops)
- ❑ ***while* loops (condition-based loops)**



Source: <https://www.google.com/imgres?imgurl=https%3A%2F%2Fthumbs.dreamstime.com%2Fb%2Fwarning-precaution-attention-alert-icon-exclamation-mark-triangle-shape-stock-vector-161619022.jpg&tbnid=cs39D65ivQdqM&vet=12ahUKEwjyqf78u8uBAxWiP94AHeaYAAAQMygFegQIARBk..i&imgrefurl=https%3A%2F%2Fwww.dreamstime.com%2Fillustration%2Falert-icon.html&docid=5Jir8W1uV2V0YM&w=800&h=800&q=red%20alert&hl=en&ved=2ahUKEwjyqf78u8uBAxWiP94AHeaYAAAQMygFegQIARBk>

“while” loops

Structure:

```
while <Boolean expression>:  
    <statements to run>
```

Adds up all the integers between 1 and N:

```
N = 10  
i = 1  
sum = 0
```

```
while i <= N:  
    sum = sum + i  
    i = i + 1
```

```
print(sum)
```

Repeat an action while
a condition is true

Explicitly increment i

while	for
while <boolean expression>: <statements to run>	for <variable> in <sequence>: <statements to run>
repeat action with a boolean expression as stop condition	repeat action in sequence
unknown number of iterations	fixed number of iterations
more general, everything with for loop can be expressed as a while loop	less error-prone when working with sequences of values
<pre>n = 1 while n < 11: if (n % 2) == 1: print(n, "is odd") else: print(n, "is even") n += 1</pre>	<pre>for n in [1,2,3,4,5,6,7,8,9,10]: if (n % 2) == 1: print(n, "is odd") else: print(n, "is even")</pre>

Quiz

A while loop repeats a block of code while a condition is true. How is this condition specified?

- A boolean expression
- An if-else statement
- With a sequence of values

“for” loops are preferable when...

- The body of the loop doesn't include any if-else statements
- Iterating over a sequence of values
- I need to explicitly specify the stopping condition of the loop

Single quotes, double quotes, and backslash

He said: "she argues: 'hello, world'"

- Put double quotes inside a single quotes;
- Put single quotes inside a double quotes;
- Use backslash to escape: if double quotes inside a double quotes, or single quotes inside a single quotes:

```
print("He said: \"she argues: 'hello, world'\"")
```

Summary:

- ❑ Introduction
- ❑ *if else* conditional statements
- ❑ *for* loops (sequence-based loops)
- ❑ *while* loops (condition-based loops)

Coding practice:

- Chapter:
- 1.3