

Topic 4: Loops, Random

Goals: By the end of this topic, we will discuss...

- loops (for, while, range), discover the properties of a loop
- random and pseudorandom numbers
- components of good documentation

Acknowledgements: These class notes build on the content of my previous courses as well as the work of R. Jordan Crouser, and Jeffrey S. Castrucci.

Loops

Goal: simplify the description of repeated blocks of code (i.e. make it shorter/easier to understand by highlighting what's being repeated and for how long).

We will start by looking at loops generally before looking at Python.

Examples:

Bake at 350 F until done.

Iterate over all the vowels.

99 bottles of beer on the wall....

This is a song that never ends...

While light is red, stop at light.

While my program has a syntax error, fix code.

For each dirty dish, wash dishes.

Counting times tables.

Group exercise:

1. Come up with all the different ways you could use a loop?
2. What are common properties? Compare and contrast these different kinds of loops.

2020 Answers:

Come up with all the different ways you could use a loop?

-
-
-
-

What are common properties? Compare and contrast these different kinds of loops.

-
-
-
-

Loops in Python

Three approaches:

- run for each item in a list (`for`)
- run a specific number of times (`for`)
- run until some condition is met (`while`)

While Loops

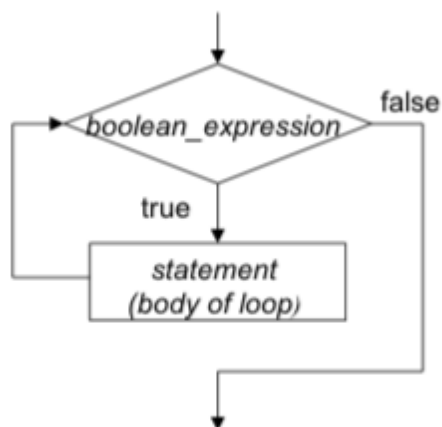
- In addition to definite loops, we may sometimes want the loop to continue until something happens
- In Python we can do this with a `while` loop, which is paired with a conditional (True/False) statement. For example:

```
x = 0
while (x < 10):
    x = x + 1
```

- `while` loops can be especially useful when combined with the `input()` function
- For example, we may want to continue asking for input until the user tells us they are done:

```
# Ask for initial input
phrase = input("Phrase (STOP to end):")

while (phrase != "STOP"):
    print("ECHO:", phrase)
    phrase = input("Phrase (STOP to end):")
```



[<https://www.cs.cmu.edu/~mrmiller/15-110/Handouts/while-4.pdf>]

Looping over a List

In Python, we use the keywords `for` and `in` to loop through a list.

- Loop over an exact set of items.

We can think of this in terms of where the variable `letter` is pointing:

```
for letter in ["A", "B", "C"]:  
    print(letter)
```

We could accomplish the same thing by writing it out as three separate assignments.
More on this when we talk about lists.

Looping over a Range

- The `range()` function lets us generate lists of integers
- Given one integer `a`, `range(a)` will generate a list starting at 0 and going up to (but not including) `a`.

For example, if we want a loop to run 5 times:

```
for i in range(5):  
    # do something
```

- Given two integers `a, b`, `range(a, b)` will generate a list starting at `a` and going up to (but not including) `b`

For example, if we want to loop over the integers from 1 to 5:

```
for i in range(1, 6):  
    # do something
```

- These values can be positive or negative (but for now, the second integer should be larger than the first)

For example, if we want to loop over the integers from -5 to 5:

```
for i in range(-5, 5):  
    # do something
```

- Given three integers `a, b, c`, calling `range(a, b, c)` will generate a list starting at `a` and going up to (but not including) `b` with step size `c`

For example, if we want the integers from 0 to 9, counting by 3s:

```
for i in range(0, 10, 3):  
    # do something
```

- If we want to count down instead of up, we can set `b < a` and use a negative step size

For example, if we want to count down from 10 to 1:

```
for i in range(10, 0, -1):  
    # do something
```

Exercise: convert °F to °C now with loops.

Use a for loop and the range() function to generate a conversion table of temperatures from °F to °C ranging from 100°F to -30°F in increments of 10°F.

This is the previous program we wrote:

```
farh = eval(input("Enter the temperature in F: "))
cel = (farh - 32) * 5//9
print(farh, " F is ", cel, " C!")
```

Tips: "{0:.1f}".format(c) will round to 1 decimal place

What does it mean for something to be **random**?

Expectation #1: even distribution

- Every value has an equal chance of being chosen
- Example: if we roll a die several times, we expect to see:
 - 1 roughly 1/6 of the time
 - 2 roughly 1/6 of the time
 - 3 roughly 1/6 of the time, etc.

On average (over a large number of samples) the distribution is roughly uniform.

Question: Is an even distribution enough?

What if the die always rolled like this?




Expectation #2: unpredictable

- Randomness is more than ensuring that every value has an equal chance of being chosen
- We also want each value to be **hard to predict**
- Specifically: seeing several values in the series (“rolls”) shouldn’t help us guess the next one

Question: Why do we care? Why do we want a random number?

Pseudorandom numbers -> "random enough"

pseu·do·ran·dom
/ˌsʊdōˈrændəm/ 

adjective

(of a number, a sequence of numbers, or any digital data) satisfying one or more statistical tests for randomness but produced by a **definite mathematical procedure**.

“random number generator” (RNG)

Translations, word origin, and more definitions

*Question: How could a **deterministic** machine generate a (seemingly) **random value**?*
We will skip the details in this class.

The random module.

- Python’s built-in random number generator (RNG) can be accessed through the random module.
- This module contains several useful functions, all of which are documented here:
<https://docs.python.org/3/library/random.html>
- This always returns a float with a value in **[0.0, 1)**
- The simplest way to get a random number is by calling the .random() function:

```
import random  
x = random.random()
```

Activity: Coin Flip

1. Use the .random() function from the random module to write a program that prints HEADS 50% of the time and TAILS the remaining 50% of the time. Try 10 coin flips.
2. Instead consider 1,000,000 coin flips. Add up (sum) the number of times it is heads and tails.



Question: What if we wanted a random float in a different range?

random floats in other ranges

- Just use math!
- Example: imagine a homework assignment is scored out of 100 points (partial points allowed, and you get 10 points for writing your name)

```
import random
# [10.0, 100.0)
score = 10.0 + random.random() * 90.0
```

Generating a random integer

- We could multiply, add and call `int(...)` to get a random integer using `.random()`, but there's no need!
- The `.randint(...)` function takes two arguments min and max, returns an integer in **[min, max] (inclusive)**:

```
import random
roll = random.randint(1, 6)
```

Different than how we have been specifying ranges... 😊

Generating a random choice

- We might want to choose an item from a list...there's a function for that!
- The `.choice(...)` function takes in a list, and returns a randomly selected element:

```
import random
flavor = ["strawberry", "chocolate", "vanilla"]
ice_cream = random.choice(flavor)
print("Today, I will have", ice_cream, "ice cream!")
```

ERROR: The `.choice(...)` function only works when given a list-like object. If you forget the square brackets, you will get a `TypeError`.

Question: What happens if we call `.choice(...)` on a string?

- Remember strings are lists of characters...
- `.choice(...)` chooses between characters

```
import random
random.choice("ABCDE")
# returns A, B, C, D, E
```

Testing Programs that Use Random

- It can be really challenging to test a program that behaves differently every time you run it.
- In order to solve this, we can tell python precisely how to generate its (not-so-random-anymore) random numbers using a parameter called a *seed*.

```
import random
random.seed(123)    # put an integer here.
for i in range(10):
    print(random.random())
```

Side Notes: Importing Multiple Modules

```
import random
import math
random_number = random.random()*100
print(math.sqrt(random_number))
```

- So far, we've always imported modules using `import <module>`
- But, two different modules may have functions of the same name.
- This causes a "name clashes" (i.e. if two functions have the same name, the second one overwrites the first)
- Instead import only the function you need:

```
from random import random
from math import sqrt
random_number = random()*100
print(sqrt(random_number))
```

- This is useful if we only need specific functions and we want to save ourselves some typing
- We can use `*` to import everything from a module :

```
from random import *
from math import sqrt
random_number = random()*100
print(sqrt(random_number))
```

- Again, just be cautious of name clashes...

While Loop Handout

Using a **while** loop, write a short code snippet to complete each task.

A. Count the number of vowels in a given string.

B. Output only the vowels included in a given string.

C. Add up all the numbers from 1 to 10.

D. Count down from 100 by 7.

E. Loop until the user enters 'q' for quit.

F. Output the times table from 1 to 12 (in a grid).

Loop (and Range) Handout

Using a `for/while` loop, write a short code snippet to complete each task.

G. Count the number of vowels in a given string.

H. Output only the vowels included in a given string.

I. Add up all the numbers from 1 to 10.

J. Count down from 100 by 7.

K. Loop until the user enters 'q' for quit.

L. Output the times table from 1 to 12 (in a grid).