Junior Knights Python I - Week 6

While Loops & Random

Agenda

For Loop Review

While Loop

Random

For Loops Review

Syntax

Defining Range

Syntax

```
for <variable> in <range>:
    stmt1
    stmt2
    ...
    stmtn
```

Note that like if statements, the for loops will only execute what is under its indentation:

```
 for <variable> in <range>:
    stmt1
    stmt2
Stmtn
```

Would result in statement n being executed outside of the for loop.

Ways to Define Range

- `range(n)` **will iterate from 0 to n-1. Note that this is not inclusive of n itself.**
- `range(a,b)`**, where a is the number you begin at, and you iterate until b-1.**
- `range(a,b,c)`**, where a and b hold the same functions as above, but c represent the "step size". This step size will be added until the value exceeds OR EQUALS b.**

While Loops

Usage

Syntax

Break, Continue, Else

Practice Problem

Usage

With the `while` loop we can execute a set of statements as long as a condition is true.

- ***While …, do …***

When is a While Loop preferred over a For Loop?

- **For loops predefine how many times a set of statements executes, restrictive**
- **While loops are used for when the amount of times is unknown**

Syntax

```
iterator = ___

while (iterator condition):
        stmt1
        stmt2
        stmt3
```

Example:

```
num = 1

while num <= 5:
print(num)
    num += 1   # Increase
```

```
while condition:
        stmt1
        stmt2
        stmt3
```

Example:

```
password = ""

while password != "magicword":
    password = input("Enter the
    password: ")

print("Correct! You may enter.")
```

Syntax

Beware of conditions that are always true, as it will cause an *endless loop*.

Example:

```
i = 4

while i < 5:
        print(“Will this ever end?”)
```

Break

With the `break` statement we can stop the loop even if the while condition is true:

Example:

```
#Exit the loop when i is 3:

i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Continue

With the `continue` statement we can stop the current iteration, and continue with the next:

Example:

```
#Continue to the next iteration if i is 3:

i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Else

With the `else` statement we can run a block of code once when the condition no longer is true:

Example:

```
#Print a message once the condition is false:

i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Live Coding

Random

Usage

Useful Methods

Usage

The random module in Python lets us make random choices, like rolling a die, flipping a coin, or picking a random number.

To use the random module, we must import it:

```
import random
```

Do this once in each program with random numbers.

```
random.seed()
```

Useful Methods

`random.randrange(start, stop, step)`

- **Returns a random number between the given range, can skip numbers using step, stop is *not* inclusive**

Example:

```
import random

num = random.randrange(1, 5, 2)
print(num)
```

`random.randint(start, stop)`

- **Returns a random number between the given range, stop *is* inclusive**

Example:

```
import random

num = random.randint(1, 10)
print(num)
```

Useful Methods

`random.choice(list)`

- **Returns a random element from the given list**

Example:

```
import random

coin = random.choice(["Heads", "Tails"])
print(coin)
```

`random.shuffle(list)`

- **Takes a list and returns the list in a random order**

Example:

```
import random

cards = ["Ace", "King", "Queen", "Jack"]

random.shuffle(cards)
```

Live Coding