Python, Day 5: Loops: Breaks, Continues and Lists

Andrew Bydlon

January 13, 2019

Loop breaks

Loops sometimes may encounter points where you want to stop them without reaching the end of the loop. This is done with a simple break statement.

You may want to exit the loop the moment a condition is met (instead of wasting CPU resources).

Syntax

```
for i in range(a,b):
    command1
    command2
    if boolean:
        break
```

Testing primality

Example (Simple prime check)

```
N = int(input("Enter a number: "))
i, prime = 2,True

while i<N:
    if N % i == 0:

        prime = False
        break # exit from for loop
    i += 1

if prime:
    print(N, "is prime")
else:
    print(N, "is not a prime. It is divisible by " + str(i) + ".")
```

breaks in nested loops

An important thing to note is that break only escapes a single loop:

Example (Specific coordinates)

```
N = int(input("Enter an integer: "))
print("Here are the coordinates (x,y) with y <= x")

for i in range(0,N+1):
    print("Start of x coordinate" i "\n **********")

    for j in range(0,N+1):
        if i<j:
            print("End of x coordinate" i "\n")
            break

        print("(", i, ", ", j, ")", sep="")
```

continue statements

Similar to the break statement, we can skip code using the continue statement.

The difference between the two operators is that continue moves past the code in the loop, but continues the loop otherwise.

Syntax

```
while boolean:
    command1
    command2
    if booleanOther:
        continue
    command 3
```

Quick example

Example (Rerunning a program alternative)

```
while True:
    value = input("Enter a number (press q at anytime to quit): ")
    if value == 'q':
        print("Exiting program.")
        break

    if not value.isdigit():
        print("Enter digits only")
        continue

    value = int(value)
    import math
    print("The exponential of ", value, "is", math.e**value)
```

Lists

Lists, as the English would entail, are collections of information. The syntax is as the sequence:

Syntax (Lists)

MyList = [item1, item2, . . . , itemN]

Example (Multiple Types)

```
> > > MyList = [4,3,2,"Hello", 6.2]
> > > for i in MyList:
            print(i, type(i))

4 <class 'int'>
3 <class 'int'>
2 <class 'int'>
Hello <class 'str'>
6.2 <class 'float'>
```

Other fun with lists

- Lists can be iterated:

SuperList = [4,3,[1,3,4],[2,6]]

- Like strings, you can pull an element of a list with [n]

Example (Pulling Elements)

> > > MyList[2]
[1,3,4]
> > > MyList[2:4]
[[1,3,4],[2,6]]

- Using the range:

Example

> > > list(range(0,10))
[0,1,2,3,4,5,6,7,8,9]

Declaring lists with for loops

Sometimes the range function is insufficient for your needs. You can also declare lists using a for statement:

Syntax

[expression for index in MySequence if Condition]

Example

```
> > > [i**2+2*i-1 for i in range(0,10)]
[-1, 2, 7, 14, 23, 34, 47, 62, 79, 98]
> > > [i**2+2*i-1 for i in range(0,10) if i%2 == 0]
[-1, 7, 23, 47, 79]
```

List Functions

As with other data types, there are many functions that you can perform on lists:

- len(MyList): finds the number of elements in the list
- sum(MyList): If the entire list is composed of numbers, produces the sum of the elements of MyList.
- max(MyList): Finds the largest entry of a list composed entirely of numbers (or strings).
- min(MyList): Finds the smallest entry.

Operations on lists

We can also use the operations (similar to strings) to produce new lists from old:

- "+": Concatenation; makes a list where entries from one list are adjoined to another.
- "*": Repetition: Adjoins the same list to itself an integer number of times.
- ">,>=,<,<=,!=,==" Tests entries of a list Lexicographically. This is similar to the dictionary ordering for strings, but allows for numeric tests as well.

Assignment 8

Ask the user for the rate r and compounding frequency n.

Then take a list of principal investments P from the user, as well as how long the user has kept the investment t. Make sure the user can input as many investments as they would like.

Using your previous code, make a list of the users Annuity at the end of their investment. Tell them what the largest, smallest, and sum of their investments was.

Upload your .py file when you finish.