

SHP Class 3

—

Lists

- In python a list is just any objects put together separated by commas in []
- Example : [1, 'two', 3.3]

- Though it is better practice to keep data types the same in a list ie: ['a', 'b', 'c']
- Lists are a way to store data, aka a data structure

Method	Description
<u>append()</u>	Adds an element at the end of the list
<u>clear()</u>	Removes all the elements from the list
<u>copy()</u>	Returns a copy of the list
<u>count()</u>	Returns the number of elements with the specified value
<u>extend()</u>	Add the elements of a list (or any iterable), to the end of the current list
<u>index()</u>	Returns the index of the first element with the specified value
<u>insert()</u>	Adds an element at the specified position
<u>pop()</u>	Removes the element at the specified position
<u>remove()</u>	Removes the item with the specified value
<u>reverse()</u>	Reverses the order of the list
<u>sort()</u>	Sorts the list

For Loop

With a for loop you know when it is going to end, like with a specific range or duration in mind.

In english, think: “For three days, I will study”

- a. A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- b. NEED and end,
- c. Remember: How python counts, programmers start with 0!!!!
- d. Declare a new variable in the same line
- e. Colon and indent

For Loop

Python syntax

```
study = 0
```

```
for i in range(3):
```

```
    study += 1
```

```
print(study)
```

Notice the indent and the ':' after the for statement

For loops

For loops are especially useful with lists!

Syntax:

```
L = [1,4,6,8,9]
```

```
for i in L:
```

```
    print(i)
```

For Loop

* ex:

```
my_list = [1, 3, 5, 7]
```

```
for item in my_list:
```

```
    print item
```

* in the example, item is a special variable that is changes in each iteration of the loop to be the "next" value, e.g. 1 the first iteration, 2 the second iteration, etc

```
thislist = ["apple", "banana", "cherry"]
```

```
print(thislist[-1])
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
```

```
print(thislist[2:5])
```

While Loop

While loops are useful when you do not know the amount of times you want something to run.

English example:

While you don't have back pain, go for runs

While Loop

- a. `while <boolean expression is true>:`
 - `#if the expression above is False, skip the whole code block`
 - `#run the code inside indent if expression above is True`
 - `#at the end of this code block, return to the top and evaluate the expression again`

Ex: print numbers 0-9

```
number = 0
```

```
while number < 10:
```

```
    print(number)
```

```
    number = number + 1
```

- b. How can we use the while loop to allow users to play our game infinitely? Think back to rock paper scissors and the fortune teller examples.

While loop

Python syntax

```
X = 0
```

```
While x < 10:
```

```
    print("hello")
```

While loops

Watch out for infinite loops!

```
x = true
```

```
While x == True:
```

```
    print( "hello")
```

Loop terms

break - breaks out of the loop once it gets to it, this can be in if statements, for loops and while loops. Breaks you out of one loop at a time.

Infinite loop - loop that never ends, not good

Nested loop - loop inside another, for inside for, while inside for... ex:

While $x < 10$:

For i in list:

print(i)

This is very useful!!! Especially when it comes to manipulating lists

Comparison Chart

BASIS FOR COMPARISON	FOR	WHILE
Declaration	<pre>for(initialization; condition; iteration){ //body of 'for' loop }</pre>	<pre>while (condition) { statements; //body of loop }</pre>
Format	Initialization, condition checking, iteration statement are written at the top of the loop.	Only initialization and condition checking is done at the top of the loop.
Use	The 'for' loop used only when we already knew the number of iterations.	The 'while' loop used only when the number of iteration are not exactly known.
Condition	If the condition is not put up in 'for' loop, then loop iterates infinite times.	If the condition is not put up in 'while' loop, it provides compilation error.
Initialization	In 'for' loop the initialization once done is never repeated.	In while loop if initialization is done during condition checking, then initialization is done each time the loop iterate.
Iteration statement	In 'for' loop iteration statement is written at top, hence, executes only after all statements in loop are executed.	In 'while' loop, the iteration statement can be written anywhere in the loop.

String Manipulation

`len(string)` - return the length of a string

`string[0]` returns the first element

NOTE: counting in python start from 0 not 1

NOTE: these also work with lists

String Manipulation

```
kanye_quote = "My greatest pain in life \
Is that I will never be able to see myself perform live."
print(kanye_quote)

print(kanye_quote.upper())
print(kanye_quote.lower())

print("Colours of the Wind".replace("Colours", "Colors"))
```

String Manipulation

- a. How do we get a specific character in a string? ***Index starts at 0**

i. `s[x]`

Ex: `s = "abcdefg"`

`s[3]`

`'d'`

- b. How do we get the length of the string?

i. `len(s)`

Ex: `s = "cat"`

`len(s)`

`3`

- c. Splicing: How do we obtain a part of the string?

i. `s[0:3]`

`s = 'hellogoodbye'`

`s[0:5]`

1. Write a Python program to remove duplicates from a list

Solution:

```
a = [10,20,30,20,10,50,60,40,80,50,40]
```

```
dup_items = set()
uniq_items = []
for x in a:
    if x not in dup_items:
        uniq_items.append(x)
        dup_items.add(x)

print(dup_items)
```

2. Write a Python program to find the second largest number in a list.

```
def second_largest(numbers):
    if (len(numbers)<2):
        return
    if ((len(numbers)==2) and (numbers[0] == numbers[1])):
        return
    dup_items = set()
    uniq_items = []
    for x in numbers:
        if x not in dup_items:
            uniq_items.append(x)
            dup_items.add(x)
    uniq_items.sort()
```