1. Recap
○○○○

2. Lists
○○○○○○○○○○○

3. For Loops
○○○○○

KOLT Python
Lists & For Loops

Fırat Tamur

Tuesday 18th February, 2020

KOÇ
UNIVERSITY
OFFICE OF LEARNING AND TEACHING

1. Recap
○○○○

2. Lists
○○○○○○○○○○○

3. For Loops
○○○○○

Agenda

1. Recap

2. Lists

3. For Loops

1. Recap
●○○○

2. Lists
○○○○○○○○○○○○

3. For Loops
○○○○○

Strings

```
my_string = 'abcde'
```

$$0\ 1\ 2\ 3\ 4$$
$$\text{'a b c d e'}$$
$$-5\ -4\ -3\ -2\ -1$$

`print(my_string[2])` $\Rightarrow$ prints c
`print(my_string[-2])` $\Rightarrow$ prints d

1. Recap
○●○○

2. Lists
○○○○○○○○○○○

3. For Loops
○○○○○

Indexing & Slicing

Access specific characters using **indexing**, i.e, `[index]`
Slice strings by using `[start:stop:step]`

```
s = 'Python'
s[1] # => 'y'
s[0:4] # => 'Pyth'
s[:3] # => 'Pyt'
s[3:] # => 'hon'
s[:] # => 'Python'
```

```
s = 'Python'
s[:5:2] # => 'Pto'
s[1:4:3] # => 'y'
s[::3] # => 'Ph'
s[::-1] # => 'nohtyP'
```

1. Recap
○○●○

2. Lists
○○○○○○○○○○○

3. For Loops
○○○○○

String Operations

```python
print('This a simple calculator program.')
number1 = input('Please enter the first number:')
number2 = input('Please enter the second number:')
print(f'{number1}+{number2} is {number1 + number2}')
```

```python
number1 = int(input('First number:'))
number2 = input('Please enter the second number:')
print(f'{number1}x{number2} is {number1 * number2}')
```

str1 + str2 $\Rightarrow$ **Concatenate** str1 and str2

str1 * n $\Rightarrow$ Repeat str1 n times.

1. Recap
○○○●

2. Lists
○○○○○○○○○○○

3. For Loops
○○○○○

While Loops

Repeat some `<expression>`s <u>as long as</u> a `<condition>` is `True`.

```
while <condition>:
    <expression>
    <expression>
    ...
```

```
x = 15
while x > 10:
    print(x)
    x-=1
```

```
counter = 11
while counter > 6:
    counter -= 1
    print(2**counter)
    counter -= 1
```

`<condition>` is only checked **<u>before</u>** each execution.

KOÇ UNIVERSITY

1. Recap
○○○○

2. Lists
●○○○○○○○○○○

3. For Loops
○○○○○

Lists

Imagine variables, but with limitless capacity...
```
sunnyside = ['Mr. Potato Head', 'Hamm',
'Buzz Lightyear', 'Slinky Dog']
```

1. Recap
OOOO

2. Lists
O●OOOOOOOOOO

3. For Loops
OOOOO

Lists

```
empty_list = []
letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5]
```

```
mixed_list = [4, 13, 'hello']
```

1. Recap
○○○○

2. Lists
○○●○○○○○○○○○

3. For Loops
○○○○○

Accessing Elements

```
values = [1, 'hello', None, [3], True]
```

```
          0      1       2      3      4
        [ 1, 'hello', None, [3], True ]
         -5    -4      -3     -2    -1
```

Use **indexing** to access and **update** elements inside list.
```
print(values[2])
values[2] = 'new value'
```

1. Recap
0000

2. Lists
000●000000000

3. For Loops
00000

Adding New Elements

Append elements at the end of a list by **append()**

```
numbers = [1, 2, 3]
numbers.append(7) # => numbers = [1, 2, 3, 7]
numbers.append(11) # => numbers = [1, 2, 3, 7, 11]

a_list = [1, 'a', 'python', 4.2]
a_list.append(3) # => a_list = [1, 'a', 'python', 4.2, 3]
a_list.append('hello')
# => a_list = [1, 'a', 'python', 4.2, 3, 'hello']
```

```
x = [1, 2, 3]
y = [4, 5]
x.append(y) # => x = [1, 2, 3, [4, 5]]
```

1. Recap
0000

2. Lists
0000●00000

3. For Loops
00000

Inspecting List Elements

Slice lists by using **[start:stop:step]**

```
x = [1, 2, 3, 4, 5]

x[2:4] # => [3,4]
x[3:4] # => [4]
x[1:-1] # => [2,3,4]
```

```
y = ['a', 'b', 'c', 'd', 'e', 'f']

y[:3] # => ['a', 'b', 'c']
y[2:] # => ['c', 'd', 'e', 'f']
y[:-1] # => ['a', 'b', 'c', 'd', 'e']
y[:] # => ['a', 'b', 'c', 'd', 'e', 'f']
```

1. Recap
0000

2. Lists
00000●000000

3. For Loops
00000

Inspecting List Elements

```
y = ['a', 'b', 'c', 'd', 'e', 'f']

y[1:5:2] # => ['b', 'd']
y[::3] # => ['a', 'd']
```

```
y = ['a', 'b', 'c', 'd', 'e', 'f']

y[::-1] # => ['f', 'e', 'd', 'c', 'b', 'a']
```

1. Recap
0000

2. Lists
○○○○○○●○○○○

3. For Loops
○○○○○

Removing An Element

Remove elements in a list by **remove()**

```
numbers = [1, 2, 3, 4]
numbers.remove(2) # => numbers = [1, 3, 4]

letters = ['a', 'b', 'c']
letters.remove('b') # => letters = ['a', 'c']

numbers_repeated = [1, 2, 5, 4, 2, 6]
numbers_repeated.remove(2) # => number_repeated = [1, 5, 4, 2, 6]

my_list = [1, 'a']
my_list.remove('b') # => ValueError
```

How to avoid ValueError? (Hint: **Branching**)

1. Recap
○○○○

2. Lists
○○○○○○○●○○○○

3. For Loops
○○○○○

in Operator

Search an operand in the specified sequence by using **in**

```
0 in [] # => False
'y' in 'Python' # => True
23 in ['hello', 40, 'a', 5] # => False
23 in ['hello', 40, 'a', 23] # => True
23 in ['hello', 40, 'a', '23'] # => False
```

- Works with both lists and strings
- Works with ranges

1. Recap
oooo

2. Lists
ooooooooo●ooo

3. For Loops
ooooo

len() Function

len() is an operator to determine the size of lists, strings, etc.

```
s = 'Python'
len(s) # => 6

my_list = [0, 1, 2, 3]
len(my_list) # => 4
```

1. Recap
0000

2. Lists
○○○○○○○○○●○○

3. For Loops
○○○○○

List Mutation

`list.append(x)`: Append x to end of the sequence
`list.insert(i, x)`: Insert x to index i
`list.pop(i=-1)`: Remove and return element at index i
`list.remove(x)`: Remove first occurrence of x
`list.extend(iterable)`: Add all elements in iterable to end of list
`list[i] = new_value`: Update value of index i with new value
`list[basic_slice] = iterable`: Change elements in basic slice with elements in iterable, sizes can be different:
`numbers[:] = []`
`list[extended_slice] = iterable`: Change elements in extended slice with elements in iterable 1-1, sizes must be equal.

1. Recap
0000

2. Lists
00000000000●0

3. For Loops
00000

Some Other List Operations

in operator: Check whether an element is in list.
3 in numbers ⇒ True
len(list): Returns the length of list(and other collections).
list.index(value, start=0, stop=len(list)):
Return first index of value.
list.count(value): Count number of occurrences of value.
list.reverse(): Reverse the list (in-place)
list.sort(): Sort list elements (in-place)

For more, type help(list) in your interactive interpreter.

1. Recap
○○○○

2. Lists
○○○○○○○○○○●

3. For Loops
○○○○○

KAHOOT TIME

To play visit the https://kahoot.it/

KOÇ
UNIVERSITY

1. Recap
OOOO

2. Lists
OOOOOOOOOOO

3. For Loops
●OOOO

range() Function

range(**start, stop, step**) is a function to create ranges

```
a = range(3) # => generates 0, 1, 2
b = range(0,3) # => generates 0, 1, 2
c = range(2,4) # => generates 2, 3
d = range(0,6,2) # => generates 0, 2, 4
0 in a # => True
1 in b # => True
4 in c # => False
2 in d # => True
6 in d # => False
```

1. Recap
OOOO

2. Lists
OOOOOOOOOOO

3. For Loops
O●OOO

For Loops

```python
for <item> in <iterable>:
    <expression>
    <expression>
    ...
```

```python
for ch in 'Python':
    print(ch)
```

```python
for num in [4,23,12,0,50]:
    print(num * 3,sep=".")
```

```python
for i in range(0,8):
    print(i)
```

1. Recap
oooo

2. Lists
ooooooooooo

3. For Loops
oo●oo

Example: Mail Sender

Fill out the attendance form:

tiny.cc/koltpython

Keyword: Gentlemen

1. Recap
OOOO

2. Lists
OOOOOOOOOOO

3. For Loops
OOO●O

Break, Continue & Pass

break immidiately terminates the closest loop

continue skips to the next iteration of the loop

```python
for i in range(0, 5):
    if i % 2 == 1:
        break
    print(i)
```

```python
for i in range(0, 5):
    if i % 2 == 1:
        continue
    print(i)
```

```python
x = 1
while x < 100:
    x *= 2
    if (x+1) % 3 == 0:
        break
    print(x)
```

```python
x = 1
while x < 100:
    x *= 2
    if (x+1) % 3 == 0:
        continue
    print(x)
```

1. Recap
OOOO

2. Lists
OOOOOOOOOOO

3. For Loops
OOOO●

Break, Continue & Pass

pass does not have an effect

```python
for letter in 'Python':
    if letter == 'y':
        pass
    else:
        print(letter)
```

```python
class CarClass():
    pass
```

- Loops, conditional statements, functions, classes etc. cannot be empty