CSIT110

Fundamental Programming with Python

Loop Statements (3)

Goh X. Y.

In this lecture

- Revising iterating through data types

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

```
for x in range(start, stop_before_this_number):
    ...
```

Data types

bool

Numbers
- **int**
- float
- complex numbers

str

list

Dict
- keys
- values

```
for x in range(start, stop_before_this_number):
    ...
```

```
x = start
x = start + 1
x = start + 2
x = start + 3
…
x  = stop_before_this_number -1
```

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

```
my_text = "the str I want to iterate"
for x in range(0, len(my_text)):
    ...
```

```
x = 0                   --> my_text[x]
x = start + 1           --> my_text[x]
x = start + 2           --> my_text[x]
x = start + 3           --> my_text[x]
…
x  = len(my_text) -1    --> my_text[x]
```

```
my_text = "the str I want to iterate"
for x in my_text:
    ...
```

```
x = my_text[0]                     x = "t"
x = my_text[1]                     x = "h"
x = my_text[2]                     x = "e"
x = my_text[3]                     x = " "
…
x  = my_text[len(my_text) -1]      x = "e"
```

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

```
my_list = [1, 2, "b", "c", []]
for x in range(0, len(my_list):
    ...
```

```
x = 0                       --> my_list[x]
x = start + 1               --> my_list[x]
x = start + 2               --> my_list[x]
x = start + 3               --> my_list[x]
…
x  = len(my_text) -1        --> my_list[x]
```

```
my_list= [1, 2, "b", "c", []]
for x in my_list:
    ...
```

```
x = my_list[0]                      x = 1
x = my_list[1]                      x = 2
x = my_list[2]                      x = "b"
x = my_list[3]                      x = "c"
…
x  = my_list[len(my_text) -1]       x = []
```

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

```
my_dict = {key1: 1,  key2: 2,  key3:"b", ...}
for x in my_dict:
    ...

 x = key1              --> y = my_dict[x]
 x = key2              --> y = my_dict[x]
 x = key3              --> y = my_dict[x]
 …
 x  = last_key         --> y = my_dict[x]
```

```
my_dict = {key1: 1,  key2: 2,  key3:"b", ...}
for x in my_dict.keys():
    ...

    x = key1        --> y = my_dict[x]
    x = key2        --> y = my_dict[x]
    x = key3        --> y = my_dict[x]
    …               …
    x  = last_key   --> y  = my_dict[x]
```

Data types

bool

Numbers
- int
- float
- complex numbers

str

list

Dict
- keys
- values

```
my_dict = {key1: 1,  key2: 2,  key3:"b", ...}
for x in my_dict:
     ...
```

```
x = key1            --> y = my_dict[x]
x = key2            --> y = my_dict[x]
x = key3            --> y = my_dict[x]
…
x  = last_key       --> y = my_dict[x]
```

```
my_dict = {key1: 1,  key2: 2,  key3:"b", ...}
for y in my_dict.values():
     ...
```

```
y = my_dict[key1]           y = 1
y = my_dict[key2]           y = 2
y = my_dict[key3]           y = "b"
…                           …
y  = my_dict[last_key]      y  = last_value
```

Are there iterators that returns two values at the same time?

Yes.

List

```
my_list = [1, 2, "b", "c", []]
for idx, val in enumerate(my_list):
```

```
idx = 0                , val = 1
idx = start + 1        , val = 2
idx = start + 2        , val = "b"
idx = start + 3        , val= "c"
…
idx  = len(my_text) -1  , val = []
```

Dic

```
my_dict = {key1: 1,  key2: 2,  key3:"b", ...}
for key, val in my_dict.items():
```

```
key = key1               , val = 1
key = key2               , val = 2
key = key3               , val = "b"
                         ...
…
key = len(my_text) -1  val = last_value
```

in keyword

for variable_name **in** range(start, stop):

for variable_name **in** <str>:

for variable_name **in** <list>:

for variable_name **in** <dict>:

sequence of data

if variable_name **in** <str>:

if variable_name **in** <list>:

if variable_name **in** <dict>:

True / False

in keyword

```
for variable_name in range(start, stop):

for variable_name in <str>:

for variable_name in <list>:

for variable_name in <dict>:
```

sequence of data

substr

```
if  variable_name in <str>:
```

- checks if substr is in the <str> obj

```
if  variable_name in <list>:
```

- checks if variable is in the <list> obj

```
if  variable_name in <dict>:
```

- checks if variable is a key in the <dict> obj

True / False

Any questions?