CISS450: Artificial Intelligence
Lecture 10: Looping

Yihsiang Liow

Agenda

- Study iteration control structure for Python: the `for` and `while` loops

Iteration

- There are two types of iteration in Python:
  - `for`
  - `while`
- Both are compound statements, i.e., of the form

```
<header>:
    <stmt>
```

while

- Format:

```
while <bool expr>:
    <stmt>
else:
    <stmt>
```

  where the `else` part is optional
- Note: C/C++, Java, etc. do not have `else` part for `while`-loop

while

- Example:

```
x = 5
sum = 0
while x > 0:
    sum += x
    x -= 1
print(sum)
```

- Example:

```
sum = 0
prompt = "Enter integer to sum (0 to end): "
i = int(input(prompt))
while i != 0:
    sum += i
    print("sum = ", sum)
    i = input(prompt)
print("Final sum is", sum)
```

while-else

- Experiment to compare `while` & `while-else`

```
x = 0
while x < 5:
    print("while ...", x)
    x += 1
print("out of while ...", x)
print
x = 0
while x < 5:
    print("while ...", x)
    x += 1
else:
    print("else ...", x)
print("out of while-else ...", x)
```

for

- Format:

```
for <target> in <seq obj>:
    <stmt>
else:
    <stmt>
```

  where the `else` part is optional
- Note: C/C++, Java, etc. do not have `else` part for `for`-loop

for

- Example:

```
sum = 0
for i in [1,2,3,4,5]:
    sum += i
```

- Example:

```
sum = 0
for i in range(1, 6):
    sum += i
```

- Example:

```
for c in "Hello, World!":
    print(c, "-")
```

for-else

- Experiment to compare `for` and `for-else`:

```
for i in [0,1,2,3,4]:
    print("in for ...", i)
print("out of for ...",i)

print
for i in [0,1,2,3,4]:
    print("in for ...", i)
else:
    print("in else ...", i)
print("out of for-else ...", i)
```

Comparing `while` and `for`

- Look at:

```
sum = 0
i = 1
While i < 5:
    sum += i
    i += 1
```

```
sum = 0
for i in [1,2,3,4]:
    sum += i
```

- If the body of the loop/iteration depends on an index and the index varies in a “predictable” way, then use `for`
- Good use of `while`: the loop terminates based on a condition that is runtime dynamic (example: based on user input)

Comparing `while` and `for`

- The following gives an infinite loop:

```
while 1:
    print("please stop ...!")
```

Comparing Python and C/C++

- `while` is similar for Python and C/C++

Python

```
while <bool expr>:
    <stmt>
```

C/C++

```
while (<bool expr>)
    <stmt>
```

- `for` is different. Look at

Python

```
for i in [1,2,3,4]:
    print(i)
```

C/C++

```
for (i=1; i<=4; i++)
    cout << i << "\n";
```

For C/C++:

- `for` has a bool condition
- `for` need not have an index variable. Example:

```
for(;;) cout << "please stop ... !";
```

Comparing Python and C/C++

- Python's `for` does not require an array index:

Python

```
names=["Joe","Mary"]

for name in names:
    print(name)
```

C/C++

```
char name[2][100]
                ={"john","doe"};
for (int i=0; i<2; i++)
  cout << name[i] << ' ';
```

Dictionaries

- Because strings and tuples are also sequence type the following works:

```
list = ['a','b','c']
tuple = ('a','b','c')
str = 'abc'
for c in list: print(c)
for c in tuple: print(c)
for c in str: print(c)
```

- You cannot iterate through a dictionary. You can however do the following:

```
dict = {1:"one", 2:"two", 3:"three"}
for key, val in dict.items():
    print(key, val)
for key in dict.keys():
    print(key, dict[key])
```

List Comprehension

- You can construct a list using ***list comprehension***
- Example:

SLOWER

```
ns = [1,2,3,4]
ms = []
for n in ns:
    ms.append(n+1)
```

FASTER

```
ns = [1,2,3,4]
ms = [n+1 for n in ns]
```

- Format:

```
[ <expr> for <target> in <seq obj> if <bool expr> ]
```

where the if part is optional

- Examples:

```
ns = [12,4,3,6,23]
print([n**2 for n in ns if n%2 == 1])
```

Miscellaneous

- Avoid index variables in `for`-loops:

BAD

```
for i in range(len(names)):
    print(names[i])
```

GOOD

```
for name in names:
    print(name)
```

- If you do need the index you can do this:

```
for index, name in enumerate(names):
    print(index, name)
```