Lecture 4: Iteration and string

Morten Rieger Hannemose, Vedrana Andersen Dahl

Fall 2023

Today's lecture

1. Iterations (ca. 15 min)
2. Strings (ca. 15 min)
3. Coding example (ca. 30 min)

Why iterate?

Iterations

- We often need to repeat the same or similar task
- Iteration: running a block of statements repeatedly
- Terminology: iteration or looping, one iteration
- Two types of iterations `for` and `while`

for statement

```python
for number in range(5):
    print(number**2)
```

- Used when you know how many times you want code to be repeated
- Later: Used to iterate over some sequence. (Later today: iterating over letters in the word.)
- Syntax for a `for` loop:
  - Keywords: `for` and `in`
  - Loop variable and a sequence
  - Function `range` generates a sequence of whole numbers, starting with 0
  - Indented body

while statement

```
a = 
while a > 10:
    a = a//2 + 2
```

- Used when you don't know how many times you want the code to be repeated
- Syntax for a `while` :
  - Keyword `while`
  - Condition: something that needs to be evaluated as either `True` or `False`
  - Indented body

Strings

```python
my_string = 'this is my string'
print(len(my_string))
print(my_string[0])
print(my_string[3])
print(my_string[4:8])
for letter in my_string:
    print(letter)
print(my_string.upper())
```

- A string is a sequence of characters
- Indexing: accessing an element of a sequence using an index
- Slicing: accessing a segment of a sequence
- A `for` loop may iterate over characters in a string
- String methods provide many useful operations. Methods are similar to functions but are used (invoked) using dot notation.

Coding example

`build_a_word.py`

Build a word

Write a function `build_a_word` that takes a string as an input. The function should print a sequence of lines. In the first line, the function should print the first letter of the string. In the second line, the function should print two first letters. The printing should continue until a whole string has been printed.

Example

Let's say you are given the string `'programming'`. We expect the print to look like:

```
p
pr
pro
prog
progr
progra
program
programm
programmi
programmin
programming
```

Coding example

`candy_exchange.py`, exam from December 2020.

Candy exchange

The owner of a candy shop came up with a plan for reducing the littering in front of the shop: “Bring back five pieces of candy wrapping, and get one candy for free!” This poses a question: If you have a certain number of candies, how many candies can you actually end up eating? Here we assume that you keep returning the wrappers as long as possible.

Problem formulation

Create a function `candy_exchange` which takes the number of candies you have as an input, and returns the number of candies you can end up eating.

Example

Let’s say you start with 36 candies.

- After eating 5 candies, and exchanging 5 wrappers for a new candy, your status is: 5 eaten and 36 - 5 + 1 = 32 candies.
- Next status: 10 eaten and 28 candies.
- Next status: 15 eaten, 24 candies.
- ...
- Next status: 40 eaten, 4 candies.
- Now you eat 4 last candies.

You can end up eating 44 candies.

Code used for coding examples

For loop and strings

```python
def build_a_word(word):
    for i in range(len(word)):
        print(word[0:i+1])
```

Also shown: `break` statement

While loop

```python
def candy_exchange(candies):
    eaten = 0
    while candies >= 5:
        candies = candies - 4
        eaten = eaten + 5
    return eaten + candies
```

Also shown: addition assignment +=

Discussed common pitfalls:

- forgetting to make sure the loop starts
- forgetting that condition is `False` when loop stops