Introduction to Programming in Python

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

A few odds and ends

Comprehensions and generator expressions

List comprehensions

- A "comprehension" in Python is a convenient way of constructing container types such as lists, dictionaries, etc.
- Lists are the most common use case.
- They look a bit like a for loop:

In [ ]:
```
[a**2 for a in range(0, 10)]
```

In [ ]:
```
[a**2 for a in range(0, 10) if a%2!=0]
```

Generator expressions

- These look like list comprehensions, but use round brackets.
- The main difference is that they don't "materialize" a list; rather, they produce their elements on demand. This saves memory.
- Example

In [ ]:

```
(a**2 for a in range(0, 10))
```

In [ ]:

```
mygen = (a**2 for a in range(0, 10))
for elem in mygen:
    print(elem)
```

Exercise

- Create a list containing the odd numbers between zero and 50, using
    - a for loop
    - a list comprehension

Exception (or error) handling

- Whenever a program needs to deal with data not under its control (user inputs, reading from files, etc.), it is a good idea to defend against illegal inputs.
- Consider the solution to the homeworks of Week 4 below.
- What happens if the user enters a string?

In [ ]:

```
import random
lower = 0
upper = 100
max_guesses = 10
to_guess = random.randint(lower, upper)
current_guess = -1
num_guesses = 0
while current_guess != to_guess:
    num_guesses += 1
    if num_guesses > max_guesses:
        print("Maximum number of guesses exceeded")
        break
    current_guess = int(input("Your guess (between {} and {}): ".format(lower, upper)))
    if current_guess < 0:
        print("Bye.")
        break
    if current_guess > to_guess:
        print("You guessed too high.")
        upper = current_guess
    elif current_guess < to_guess:
        print("You guessed too low.")
        lower = current_guess
else:
    print("You got it!")
```

- Often, rather than just throwing an error and terminating the program, it is desirable to have the program deal with the error.
- This is what exception/error handling is for.
- Basic syntax:

In [ ]:

```
try:
    num_str = input("Please enter an integer: ")
    num = int(num_str)
except:
    print("You entered {}, which cannot be converted to an integer".format(num_str))
```

Exercise

- Use exception handling to defend against illegal inputs in the number guessing game above.

Recommended reading

- https://www.geeksforgeeks.org/python-list-comprehensions-vs-generator-expressions/

Further reading

- https://python-course.eu/python-tutorial/errors-and-exception-handling.php

Homework

- https://towardsdatascience.com/beginner-to-advanced-list-comprehension-practice-problems-a89604851313 Exercises 1-3