3. Advanced Python Concepts

Built-in Functions, Conditional Expressions, List and Dict Comprehension, File IO, Exception-Handling

Built-In Functions: Enumerate with Indices

Sometimes, one wants to iterate through a list, including the index of each element. This works with **`enumerate( ... )`**

```
data = [ 'Spam', 'Eggs', 'Ham' ]

for index, value in enumerate(data):
  print(index, ":", value)
```

Output:

```
0 : Spam
1 : Eggs
2 : Ham
```

Built-In Functions: Combining lists

There is a simple possibility to combine lists element-wise (like a zipper!):

`zip( ... )`

```
places = [ 'Zurich', 'Basel', 'Bern']
plz = [ 8000, 4000, 3000, ]

list(zip(places, plz)
# [('Zurich', 8000), ('Basel', 4000), ('Bern', 3000)]

dict(zip(places, plz)
# {'Zurich': 8000, 'Basel': 4000, 'Bern': 3000}
```

Conditional Expressions

In Python, the value of an expression can depend on a condition (as part of the expression!)

Example: Collaz Sequence

```
while a != 1:
    a = a // 2 if a % 2 == 0 else a * 3 +1
```

Example: Text formatting

```
print('I see', n, 'mouse' if n ==1 else 'mice')
```

List Comprehension

- Python provides a convenient way of creating lists declaratively
- Similar technique to ‘map’ and ‘filter’ in functional languages

Example: Read-in a sequence of numbers

```
line = input('Enter some numbers: ')
s_list = line.split()
n_list = [ int(x) for x in s_list ]
```

The same combined in one expression

```
n_list = [ int(x) for x in input('Enter some numbers: ').split() ]
```

List Comprehension

Example: Eliminate whitespace in front and at the back

```
line = [ ' some eggs  ', ' slice of ham ', ' a lot of spam ' ]
cleaned = [ item.strip() for item in line ]

# cleaned == [ 'some eggs', 'slice of ham', 'a lot of spam' ]
```

Dict Comprehension

- Like with lists, but with key/value pairs

Example: extract data from a dict

```
data = {
    'Spam' : { 'Amount' : 12, 'Price': 0.45 },
    'Eggs' : { 'Price': 0.8 },
    'Ham'  : { 'Amount': 5, 'Price': 1.20 }
}

total_prices = { item : record['Amount'] * record['Price']
    for item, record in data.items()
    if 'Amount' in record }

# total_prices == {'Spam': 5.4, 'Ham': 6.0}
```

File IO

- Files can be opened with the command **`open`**
- To automatically close files afterwards, this must happen in a **`with`** block

Example: Read CSV file

```
import csv

with open('times.csv', mode='r') as csv_file:
    csv_lines = csv.reader(csv_file)
    for line in csv_lines:
      # do something for each record
```

Writing works similarly. See Python documentation.

Exception Handling

Given the following code:

```
x = int(input('A number please: '))
```

If no number is entered, the program crashes:

```
Traceback (most recent call last):
  File "main.py", line 1, in <module>
    x = int(input('A number please: '))
ValueError: invalid literal for int() with base 10: 'a'
```

We can catch this error and react accordingly.

Exception Handling

```
try:
  x = int(input('A number please: '))
except ValueError:
  print('Oh boy, that was no number...')
  x = 0
print('x:', x)
```

Output, if `spam` is entered instead of a number:

```
Oh boy, that was no number...
x: 0
```