



Exercise Session 2 - First Python Programs

Computer Science II

Wednesday, March 1, 2023

Program Today

Repetition of Course Content

Exceptions

In-class Exercises

Homework

1. Repetition of Course Content

Variables

Dynamic typing:¹ Types exist in Python. They are assigned during program run, not defined by the programmer.

Python

```
i = 1
```

```
d = 1.0
```

```
c = 'a'
```

```
b = True
```

C++

```
int i = 1;
```

```
double d = 1;
```

```
char c = 'a';
```

```
bool b = true;
```

¹This topic will be covered in depth later in the lecture.

Containers

Sequences (ordered)

- tuple
- list
- range
- string

Collections (unordered)

- set
- dictionary

Container Operations

Number of elements

```
len(c)
```

Does c contain x?

```
b = x in c
```

Iteration over c

```
for x in c:  
    print(x)
```

Container Operations

Python

Number of elements

```
len(c)
```

Does c contain x?

```
x in c
```

Iteration over c

```
for x in c:  
    print(x)
```

C++

```
c.size();
```

```
std::find(c.begin(), c.end(), x);
```

```
for(int i=0;i<c.size();i++)  
    std::cout << c[i] << "\n";
```

Quiz

For all questions on this slide, assume:

`c =`

1	3.14	7	'a'	True
0	1	2	3	4

What is the output of the following commands?

```
len(c)
```

```
2 in c
```

```
for x in c:  
    print(x)
```


Sequences

- **tuple** (*all objects, immutable*)

```
t = (0, 'a', 3.3)
```

- **list** (*all objects, mutable*)

```
l = [1.0, 5, 'hi', -2]
```

- **range** (*numbers, immutable*)

```
r = range(1,8,2)
```

- **string** (*characters, immutable*)

```
s = "ETH"
```

t =

0	'a'	3.3
0	1	2

l =

1.0	5	'hi'	-2
0	1	2	3

r =

1	3	5	7
0	1	2	3

s =

'E'	'T'	'H'
0	1	2

Sequence Operations

- Subscript operator

```
s[i]
```

- Enumeration

- Combine each element with its position.

```
for (i,x) in enumerate(s):  
    print(i,x)
```

- Zip

- Combine two sequences together.

```
z = zip(s,t)  
l = list(z)
```

Enumeration

s =

2	3	5	8	13
0	1	2	3	4

```
for (i,x) in enumerate(s):  
    print(i,x)
```

```
0 2  
1 3  
2 5  
3 8  
4 13
```

Zip

s =	<div>2</div> <div>0</div>	<div>3</div> <div>1</div>	<div>5</div> <div>2</div>	<div>8</div> <div>3</div>	<div>13</div> <div>4</div>
t =	<div>3</div> <div>0</div>	<div>6</div> <div>1</div>	<div>9</div> <div>2</div>	<div>12</div> <div>3</div>	<div>15</div> <div>4</div>

```
z = zip(s,t)  
l = list(z)
```

l =	<div>(2,3)</div> <div>0</div>	<div>(3,6)</div> <div>1</div>	<div>(5,9)</div> <div>2</div>	<div>(8,12)</div> <div>3</div>	<div>(13,15)</div> <div>4</div>
-----	-------------------------------	-------------------------------	-------------------------------	--------------------------------	---------------------------------

Slicing

Selecting a subsequence according to the following rules:

- Start at **start**, End **before stop**, Step size **step**

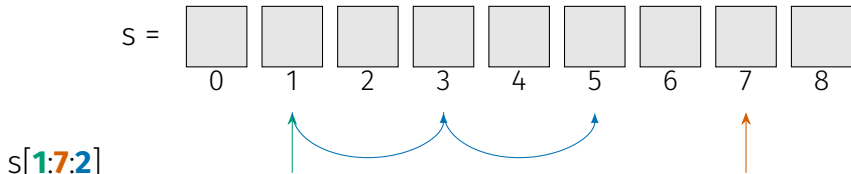
```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Slicing

Selecting a subsequence according to the following rules:

- Start at **start**, End **before stop**, Step size **step**

```
s[start:stop:step]  
s[start:stop] #step = 1  
s[:stop:step] #start = 0  
s[start::step] #stop = len(s)
```



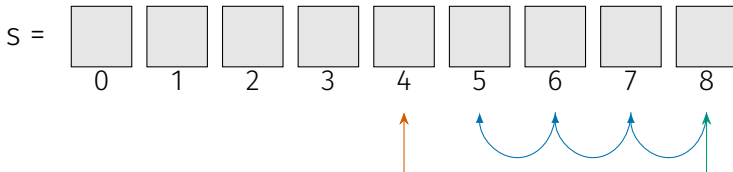
Slicing

Selecting a subsequence according to the following rules:

- Start at **start**, End **before stop**, Step size **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Negative **step**: go backward.



`s[8:4:-1]`

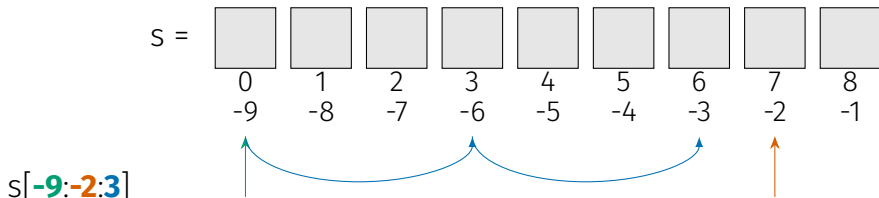
Slicing

Selecting a subsequence according to the following rules:

- Start at **start**, End **before stop**, Step size **step**

```
s[start:stop:step]
s[start:stop] #step = 1
s[:stop:step] #start = 0
s[start::step] #stop = len(s)
```

Negative **start**, **stop**: use negative indexing.



Slicing: Quiz

On this slide, assume:

```
s = [1, 2, 3, 5, 8, 13, 21, 34, 55]
```

What is the output of the following code?

```
s[3::5]
```

How would you slice sequence s to produce the following output?

```
[34, 8, 2]
```

Slicing: Quiz

Let us have a sequence s:

```
s = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O']
```

How would you slice to produce the following lists?

```
['E', 'F', 'G', 'H', 'I']
```

```
['L', 'K', 'J', 'I']
```

```
['C', 'H']
```

```
['O', 'L', 'I', 'F']
```

Range

A sequence that starts at **start**, ends **before stop** with step size **step**.

```
range(start, stop, step)
range(start, stop) #step = 1
range(stop) #start = 0, step = 1
```

Range is often used in for loops:

Python

```
for i in range(a, b, c):
    do_something
```

C++

```
for(int i=a; i<b; i+=c)
    do_something;
```

Quiz

What is the output of the following code?

```
tuple(range(3,15,4))
```

```
tuple(range(19,2,-2)[2:7:3])
```

How would you generate the following output using one range command?
Can you think of another range command doing the same? How many are there?

```
(2019, 2023, 2027)
```

Quiz

How would you generate the following output using one range command?

```
[-12, -6, 0, 6, 12]
```

```
[8, 4, 0, -4]
```

How would you slice `range(15,-15,-3)` to get the following output?

```
[-9, -3, 3, 9]
```

Operations on List

- Change an element

```
l[i] = val
```

- Append an element

```
l.append(val)
```

- Remove an element

```
del l[i]
```

- Reverse the list

```
l.reverse()
```

- List of k elements with value val

```
l = [val] * k
```

Quiz

What does list l look like after each step?

```
l = [0] * 4  
l[1] = 3  
l.append(5)  
l.reverse()  
del l[3]
```

2. Exceptions

Example

- Exceptions are raised when the program is syntactically correct but the code resulted in an error.
- Some of the standard exceptions which are most frequent include **IndexError**, **ImportError**, **IOError**, **ZeroDivisionError**, **TypeError**, and **FileNotFoundError**.
- Following example raises a **ZeroDivisionError** exception, as we are trying to divide a number by 0.

```
a = 1000  
b = a / 0  
print(b)
```

```
ZeroDivisionError  
Cell In[1], line 2  
      1 a = 1000  
----> 2 b = a / 0  
      3 print(b)
```

Traceback (most recent call last)

ZeroDivisionError: division by zero

Exception Handling

- We can use **try** and **except** clauses to handle exceptions.
- A try statement can have more than once except clause, to specify handlers for different exceptions.
- However, at most one handler will be executed.

```
a = [1, 2, 3]
try:
    print("Second element = %d" %(a[1]))
    # Throws error since there are only 3 elements in array
    print("Fourth element = %d" %(a[3]))
except IndexError:
    print("An error occurred")
```

Output:

```
Second element = 2
An error occurred
```

Finally

- **finally** defines code that is always executed after a try and except block, independently if an exception is raised or not.

```
try:
    k = 5//0 # raises divide by zero exception.
    print(k)

# handles zerodivision exception
except ZeroDivisionError:
    print("Can't divide by zero")

finally:
    # this block is always executed
    # regardless of exception generation.
    print("This is always executed")
```

Output:

```
Can't divide by zero
This is always executed
```

User-Defined Exceptions

- You can create your own exceptions by creating a new class.
- New exceptions must derive from the **Exception** class.

```
class MyError(Exception):  
    # Constructor or Initializer  
    def __init__(self, value):  
        self.value = value  
    # __str__ is to print() the value  
    def __str__(self):  
        return(repr(self.value))  
  
try:  
    raise(MyError(3*2))  
# Value of Exception is stored in error  
except MyError as error:  
    print("A New Exception occurred: ", error.value)
```

Output:

A New Exception occurred: 6

3. In-class Exercises

Reading user input

```
word = input("Enter a word : ")
```

- This code writes a text "Enter a word : " on a console and waits for user input.
- After the user enters some text, it is stored into variable `word` as data type string.

Reading user input in a loop

```
word = input("Enter a word : ")
again = True
while again:
    #Do something with word...
    word = input("Enter a word (or just <ENTER> to stop): ")
    again = len(word) > 0
```

- This code sequentially reads strings from user, and processes them.
- If the user enters an empty string, the program terminates.

In-class exercise: Palindrome

A **palindrome** is a word that is spelt the same way backwards and forwards.

Write a python program that:

- Sequentially reads words (possibly containing spaces) from user input.
- For each word, the program prints whether the word is a palindrome.
- If the user enters an empty string, the program terminates.

Go to CodeExpert - Code Examples - Exercise 2 - In-class

Hint: a string is a sequence. All sequence operations can be applied to it.

In-class exercise: Count of numbers above average

Write a python program with the following input, and output:²

Input: A list `s` of numbers.

Output: The count of numbers in list `s` that are strictly larger than the average value.

Example: `s = [1,1,2,3,4,1]`

The average value in list `s` is equal to 2. There are two numbers in `s` that are larger than 2: 3, and 4. Therefore, the output should be 2.

²Do this exercise if there is spare time.

4. Homework

Exercise 1: Python I

On <https://expert.ethz.ch/mycourses/SS23/mavt2/exercises>

- Sum and Maximum
- List Comprehension
- Dict Comprehension
- Crops & Dictionaries

Due date: Monday 06.03.2023, 20:00 CET

NO HARDCODING

Questions?