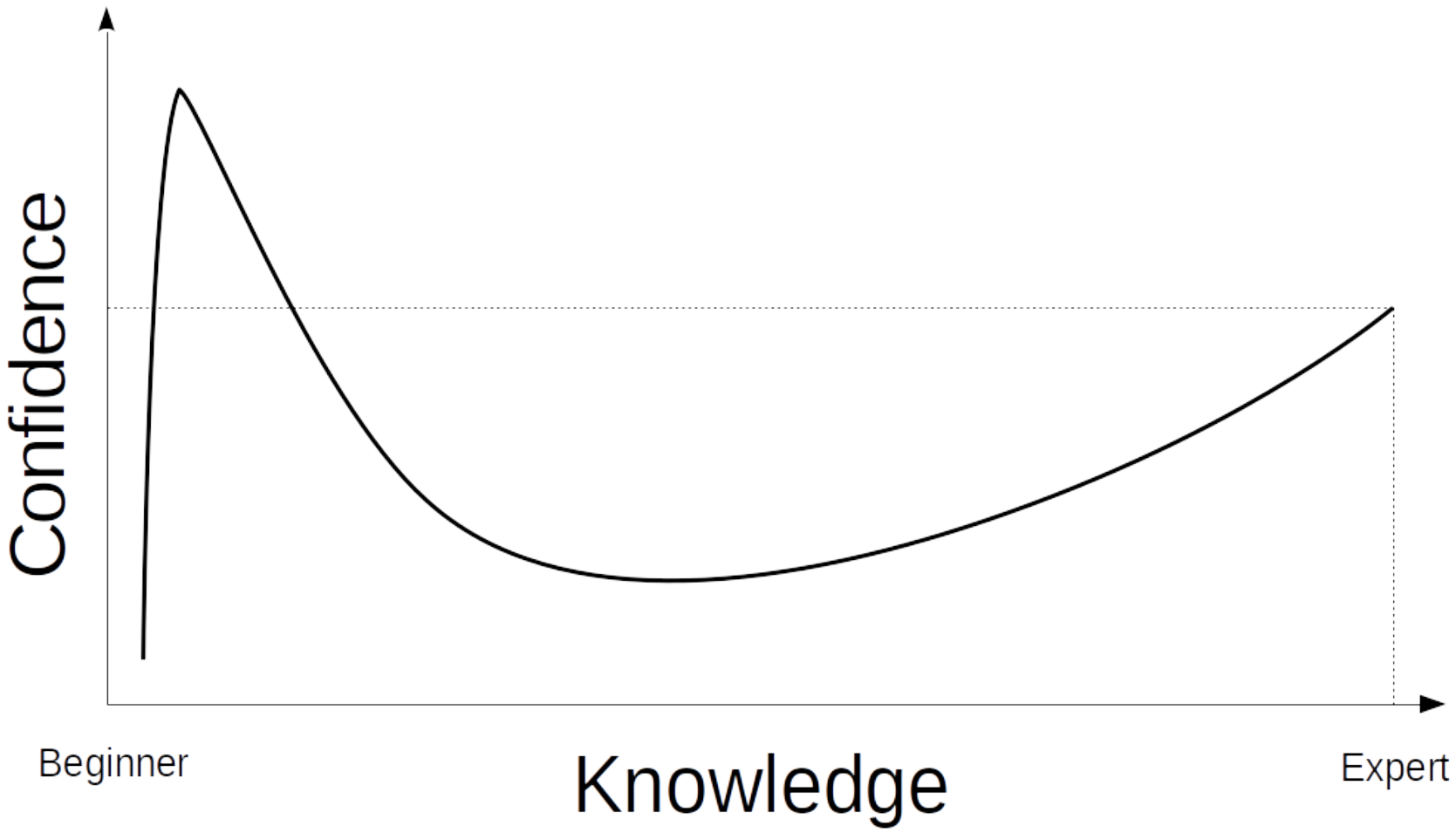
Dyys mypkukd!

Good morning!

The seminar so far

Date	Topic	Contents
1/7/2019	1: Meet your (other) computer!	Terminals, files and folders
3/7/2019	2: Learning Python	Python data types and logic
5/7/2019	3: Using Python	Lists and loops
8/7/2019	4: Solving actual problems	More lists, exceptions and pseudo code
10/7/2019	5: Python building blocks	Functions, modules, sets and dictionaries
13/7/2019	6: Working with files and objects	Reading and writing to files, OOP
15/7/2019	7: Working with data	Documentation, testing and excel data



Documentation

Ways to document your code, by writing human readable text that describes:

1. the *purpose* of your variable, function, class or method
2. the input parameters of your function, constructor or method
3. the output result of your function or method

Testing

Programs that test that your program works.

Using programs from programs

Programs are not just programs. Programs exist to be used!

- CLI programs
- Testing

Example: Turtle game with program installer

Object orientation with Excel

- Object-oriented programming (OOP)
  - Class vs. object
  - Properties and methods
- Heavily used in the real world
  - Excel (with `openpyxl`)
  - Loads of other libraries

Learning outcomes

The expected outcomes for the seminar that you will be tested for.

The student is able to

- Analyse a given, simple computation task such as manipulation of a text-based database or external hardware device to the extent of designing a programmatic solution and implementing it in a modern, text-based, domain-neutral programming language
- Test the correctness of a piece of code
- Write program documentation
- Express functionality in terms of abstract data type or application programming interface
- Use text-based tools of program development, including an editor, command-line tools, and a version control system

- Reason about the computation complexity of an algorithm

IMPORTANT!!!

Bring a *billede ID*, kørekort, your ITU card, etc. to the test!

Otherwise you cannot enter the room and participate in it...

Handling exceptions and getting the exception object

In [1]:

```
1 print('Hej ' + 3)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-1-f8f136489f52> in <module>
----> 1 print('Hej ' + 3)

TypeError: can only concatenate str (not "int") to str
```

In [2]:

```
try:
    print('Hej ' + 3)
except:
    print("Don't you remember that you cannot add integers and strings?")
```

```
Don't you remember that you cannot add integers and strings?
```

In [4]:

```
try:
    print('Hej ' + 3)
except Exception as e:
    print(e)
    print(type(e))
```

```
can only concatenate str (not "int") to str
True
```

Reading a file and skipping some lines with the `next` function

This is new, we did not cover that yet!

In [ ]:

```
filename = 'bones_in_london.txt'
with open(filename) as fp:
    for line in fp:
        print(line.strip())
```

The first 18 lines of that file do not seem to be part of the actual book but more a header, i.e., meta-information that you might want to skip when reading the file.

In [ ]:

```
filename = 'bones_in_london.txt'
with open(filename) as fp:
    next(fp)
    for line in fp:
        print(line.strip())
```

In [ ]:

```
filename = 'bones_in_london.txt'
with open(filename) as fp:
    for _ in range(18):
        next(fp)
    for line in fp:
        print(line.strip())
```

What was this `sys.argv` again?

> The command line arguments will be stored in the variable `sys.argv`. [...] The first item in the `sys.argv` list should always be a string containing the program's filename [...], and the second item should be the first command line argument.
>
> Al Sweigart, "*Automate the Boring Stuff with Python*"

If still in doubt, check p.136 in chapter 6 in "*Automate the Boring Stuff with Python*"

In essence it is a *variable* in the `sys` module containing a *list* of strings, the arguments given to this Python program.

The `sorted` function, what does it do?

Did you read its documentation?

In [9]:

```
1 print(help(sorted))
```

```
Help on built-in function sorted in module builtins:

sorted(iterable, /, *, key=None, reverse=False)
    Return a new list containing all items from the iterable in asce
nding order.

    A custom key function can be supplied to customize the sort orde
r, and the
    reverse flag can be set to request the result in descending orde
r.

None
```

But how does the computer actually sort???

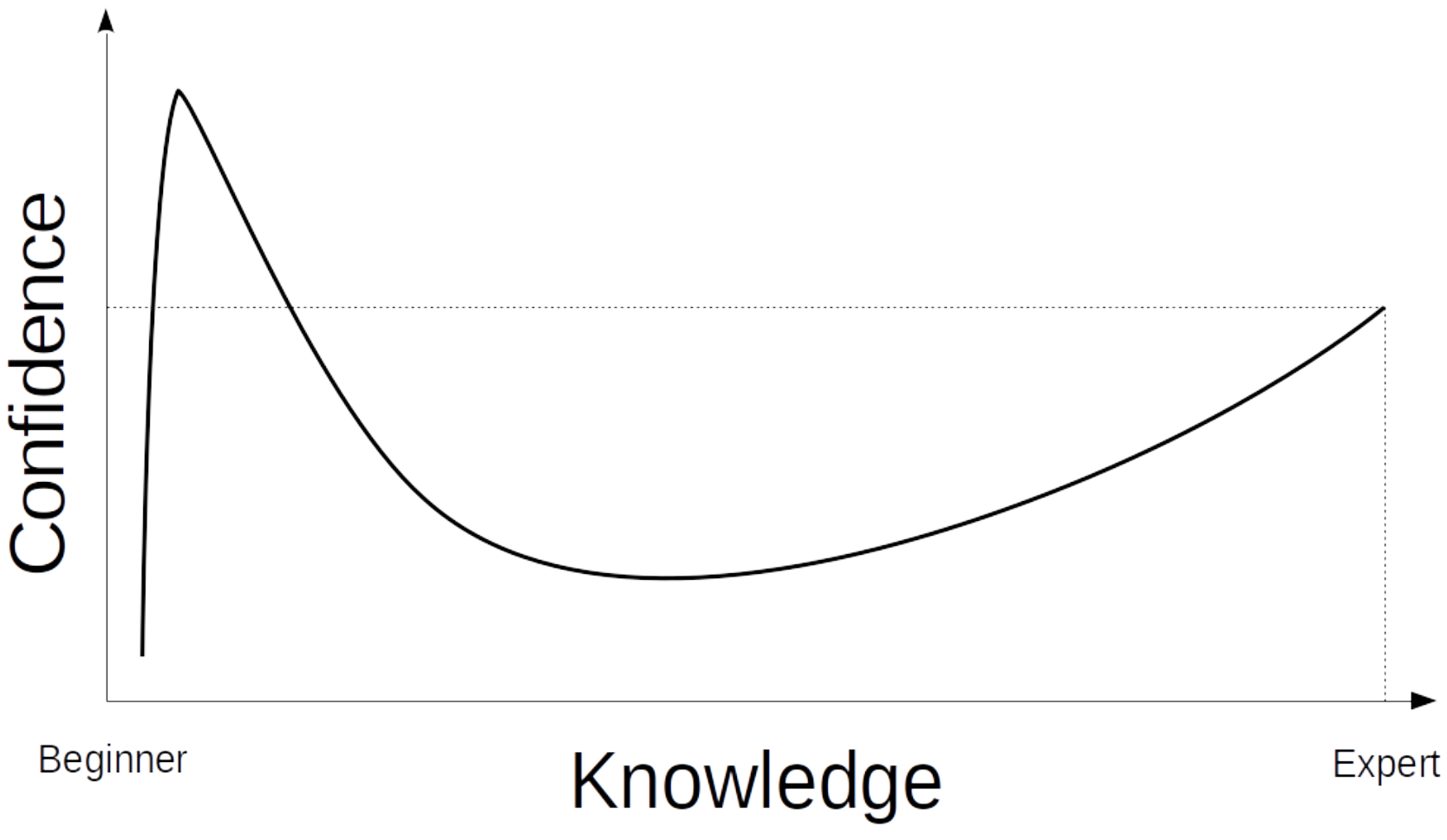
More on sorting in some minutes :)

Dyys mypkukd!

Good morning!

The seminar so far

Date	Topic	Contents
1/7/2019	1: Meet your (other) computer!	Terminals, files and folders
3/7/2019	2: Learning Python	Python data types and logic
5/7/2019	3: Using Python	Lists and loops
8/7/2019	4: Solving actual problems	More lists, exceptions and pseudo code
10/7/2019	5: Python building blocks	Functions, modules, sets and dictionaries
13/7/2019	6: Working with files and objects	Reading and writing to files, OOP
15/7/2019	7: Working with data	Documentation, testing and excel data



Documentation

Ways to document your code, by writing human readable text that describes:

1. the *purpose* of your variable, function, class or method
2. the input parameters of your function, constructor or method
3. the output result of your function or method

Testing

Programs that test that your program works.

Using programs from programs

Programs are not just programs. Programs exist to be used!

- CLI programs
- Testing

Example: Turtle game with program installer

Object orientation with Excel

- Object-oriented programming (OOP)
  - Class vs. object
  - Properties and methods
- Heavily used in the real world
  - Excel (with `openpyxl`)
  - Loads of other libraries

Learning outcomes

The expected outcomes for the seminar that you will be tested for.

The student is able to

- Analyse a given, simple computation task such as manipulation of a text-based database or external hardware device to the extent of designing a programmatic solution and implementing it in a modern, text-based, domain-neutral programming language
- Test the correctness of a piece of code
- Write program documentation
- Express functionality in terms of abstract data type or application programming interface
- Use text-based tools of program development, including an editor, command-line tools, and a version control system

- Reason about the computation complexity of an algorithm

IMPORTANT!!!

Bring a *billede ID*, kørekort, your ITU card, etc. to the test!

Otherwise you cannot enter the room and participate in it...

Handling exceptions and getting the exception object

In [1]:

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-1-f8f136489f52> in <module>
----> 1 print('Hej ' + 3)

TypeError: can only concatenate str (not "int") to str
```

In [2]:

Don't you remember that you cannot add integers and strings?

In [4]:

```
can only concatenate str (not "int") to str
True
```

Reading a file and skipping some lines with the `next` function

This is new, we did not cover that yet!

In [ ]:

The first 18 lines of that file do not seem to be part of the actual book but more a header, i.e., meta-information that you might want to skip when reading the file.

In [ ]:

In [ ]:

What was this `sys.argv` again?

> The command line arguments will be stored in the variable `sys.argv`. [...] The first item in the `sys.argv` list should always be a string containing the program's filename [...], and the second item should be the first command line argument.
>
> Al Sweigart, "*Automate the Boring Stuff with Python*"

If still in doubt, check p.136 in chapter 6 in "*Automate the Boring Stuff with Python*"

In essence it is a *variable* in the `sys` module containing a *list* of strings, the arguments given to this Python program.

The `sorted` function, what does it do?

Did you read its documentation?

In [9]:

```
Help on built-in function sorted in module builtins:

sorted(iterable, /, *, key=None, reverse=False)
    Return a new list containing all items from the iterable in asce
nding order.

    A custom key function can be supplied to customize the sort orde
r, and the
    reverse flag can be set to request the result in descending orde
r.

None
```

But how does the computer actually sort???

More on sorting in some minutes :)