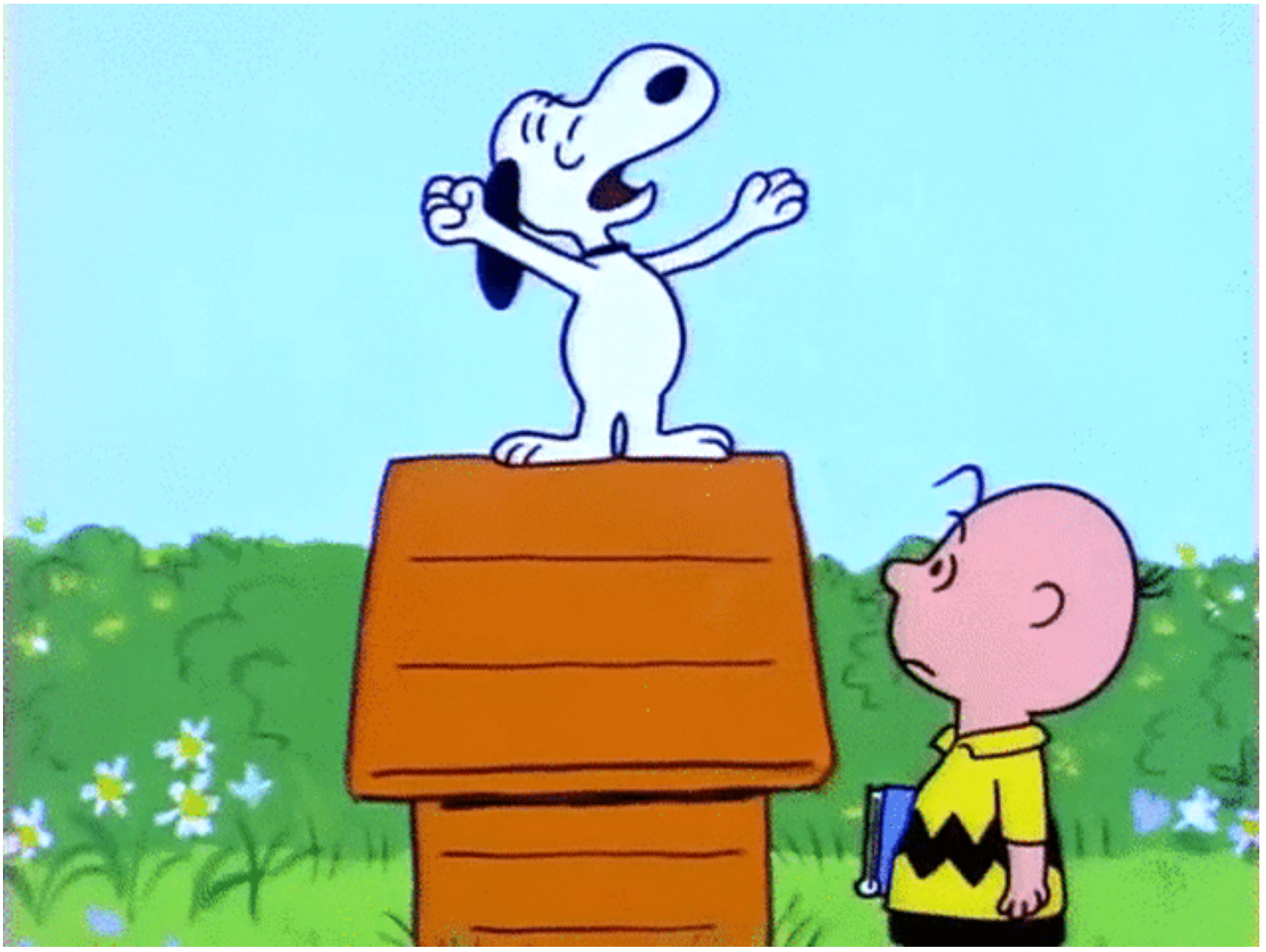
I still do not understand completely what the issue was with LearnIT yesterday.

However, I just reopened the hand-in and the group selection item (in case it is needed).

Can you please try to hand-in again. We just tried it with one of you and it should work now.

Please hand-in before 10am please.

Good morning!



IMPORTANT!!!

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

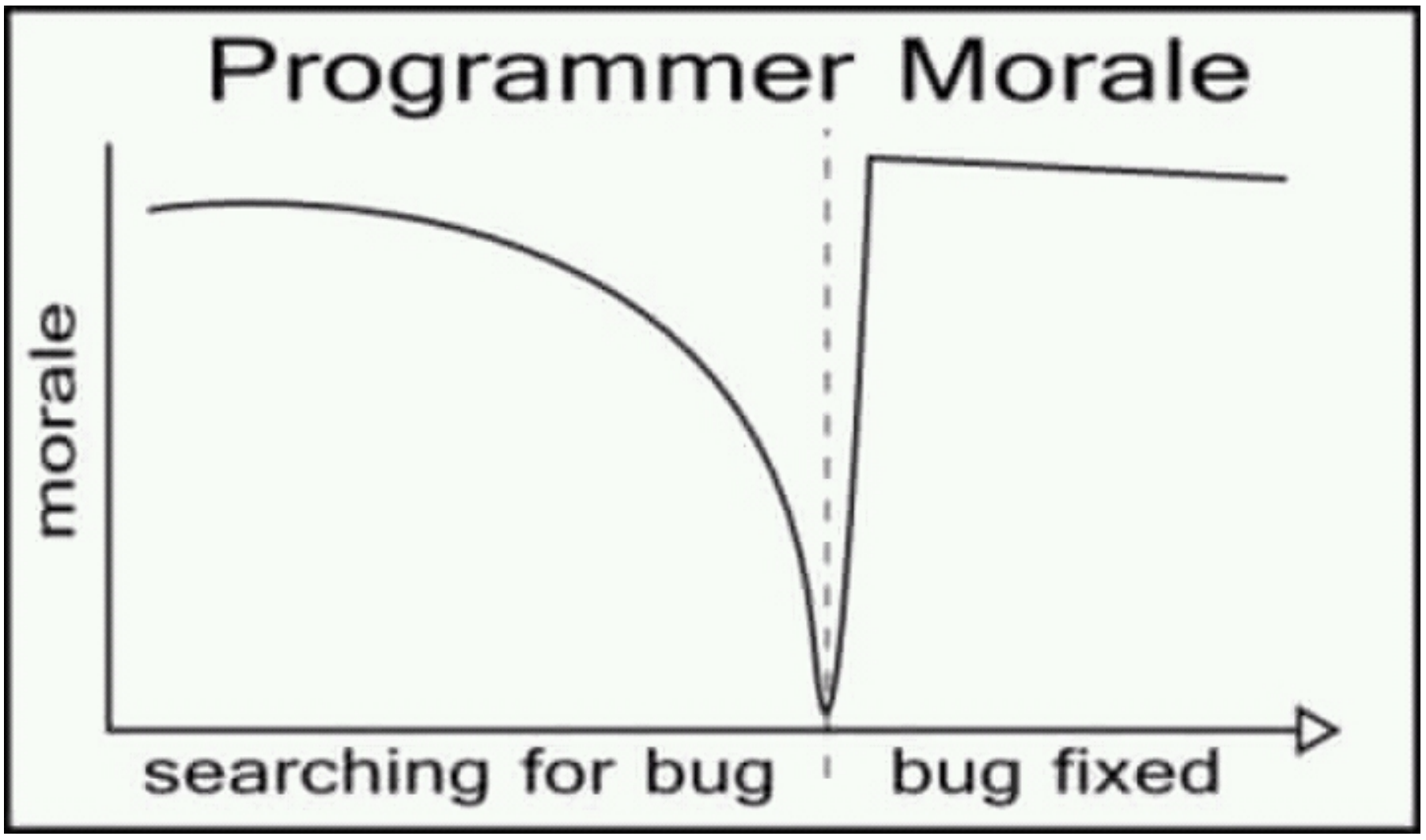
Otherwise you cannot enter the room and participate in it...

Assignment Feedback

Hand-in Technical Issue

We are sorry! We do not know what was going on.

Likely you experienced something like this during the assignment



Turtle Runner

Do you have an idea of why I am thinking this was one of the coolest exercises in the entire seminar?

In [ ]:

```
import sys
from turtle import Turtle, done


def do_line(turtle, command, value):
    if command == 'Walk':
        turtle.forward(value)
    if command == 'Turn':
        turtle.right(value)


def main(path_to_file):
    dave = Turtle()
    dave.shape('turtle')
    with open(path_to_file, 'r') as script:
        for line in script:
            command, value = line.split(' ')
            value = int(value)
            do_line(dave, command, value)


if __name__ == '__main__':
    main(sys.argv[1])
    done()
```

Turtle Geometry

In [ ]:

```
from turtle import Turtle


class GeometryTurtle(Turtle):
    def make_rectangle(self, width, height):
        for _ in range(2):
            self.forward(width)
            self.left(90)
            self.forward(height)
            self.left(90)

    def make_square(self, width):
        self.make_rectangle(width, width)

    def make_star(self, length):
        for _ in range(5):
            self.forward(length)
            self.left(144)

    def make_triangle(self, length):
        for _ in range(3):
            self.forward(length)
            self.left(120)


my_turtle = GeometryTurtle()
my_turtle.make_square(50)
```

Loops

`for` loops and `while` loops

They can express the same. Remember Morten's slide!: This:

In [19]:

```
my_list = [1, 'elephant', 4, 'rats']

index = 0
while index < len(my_list):
    element = my_list[index]
    print(element)
    index += 1
```

```
1
elephant
4
rats
```

is the same as

In [ ]:

```
my_list = [1, 'elephant', 4, 'rats']

for element in my_list:
    print(element)
```

But I would like to write a `for` loop but still need the index

You may want to use the `enumerate` function.

In [17]:

```
1 print(help(enumerate))
```

```
Help on class enumerate in module builtins:

class enumerate(object)
 |  enumerate(iterable, start=0)
 |
 |  Return an enumerate object.
 |
 |    iterable
 |      an object supporting iteration
 |
 |  The enumerate object yields pairs containing a count (from start
, which
 |  defaults to zero) and a value yielded by the iterable argument.
 |
 |  enumerate is useful for obtaining an indexed list:
 |      (0, seq[0]), (1, seq[1]), (2, seq[2]), ...
 |
 |  Methods defined here:
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __next__(self, /)
 |      Implement next(self).
 |
 |  __reduce__(...)
 |      Return state information for pickling.
 |
 |  ----------------------------------------------------------------
------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate
signature.

None
```

In [27]:

```
my_list = [1.0, 'elephant', 4.0, 'rats']

for idx, element in enumerate(my_list):
    print(idx)
    print(element)
    print(my_list[idx - 1])
```

```
0
1.0
rats
1
elephant
1.0
2
4.0
elephant
3
rats
4.0
```

The last basic data type: `tuple`

A list:

In [5]:

```
my_list = [1, 'elephant', 4, 'rats']
```

A tuple, is an immutable list

In [28]:

```
my_list = [1, 'elephant', 4, 'rats']
my_tuple = (1, 'elephant', 4, 'rats')

print(my_list[1])
print(my_tuple[2])
```

```
elephant
4
```

In [29]:

```
my_list = [1, 'elephant', 4, 'rats']
my_tuple = (1, 'elephant', 4, 'rats')

my_list[1] = 'mouse'
print(my_list[1])
my_tuple[1] = 'mouse'
print(my_tuple[2])
```

```
mouse

---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-29-588241f7057d> in <module>
      4 my_list[1] = 'mouse'
      5 print(my_list[1])
----> 6 my_tuple[1] = 'mouse'
      7 print(my_tuple[2])

TypeError: 'tuple' object does not support item assignment
```

The `type` function revisited

Now, that we started with object-oriented programming, the `type` function likely makes more sense. You give it an object and it tells you of which class (type) that object is an `instanceof` .

In [12]:

```
print(help(type))
```

```
Help on class type in module builtins:

class type(object)
 |  type(object_or_name, bases, dict)
 |  type(object) -> the object's type
 |  type(name, bases, dict) -> a new type
 |
 |  Methods defined here:
 |
 |  __call__(self, /, *args, **kwargs)
 |      Call self as a function.
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __dir__(self, /)
 |      Specialized __dir__ implementation for types.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
```

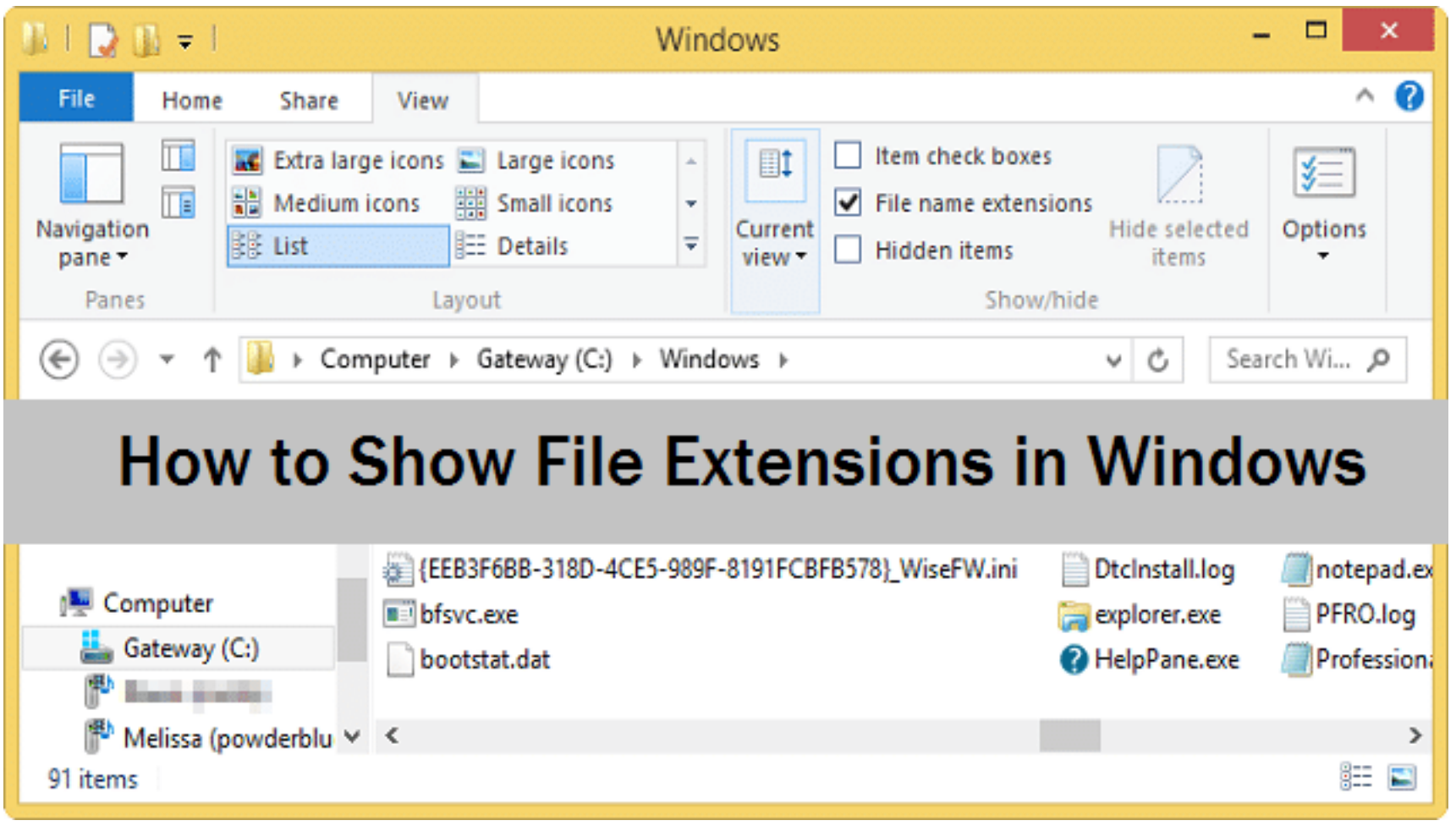
```
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signatur
e.
 |
 |  __instancecheck__(self, instance, /)
 |      Check if an object is an instance.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __sizeof__(self, /)
 |      Return memory consumption of the type object.
 |
 |  __subclasscheck__(self, subclass, /)
 |      Check if a class is a subclass.
 |
 |  __subclasses__(self, /)
 |      Return a list of immediate subclasses.
 |
 |  mro(self, /)
 |      Return a type's method resolution order.
 |
 |  ----------------------------------------------------------------
------
 |  Class methods defined here:
 |
 |  __prepare__(...)
 |      __prepare__() -> dict
 |      used to create the namespace for the class statement
 |
 |  ----------------------------------------------------------------
------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate
signature.
 |
 |  ----------------------------------------------------------------
------
 |  Data descriptors defined here:
 |
 |  __abstractmethods__
 |
 |  __dict__
 |
 |  __text_signature__
 |
 |  ----------------------------------------------------------------
------
 |  Data and other attributes defined here:
 |
 |  __base__ = <class 'object'>
 |      The most base type
```

```
|
|   __bases__ = (<class 'object'>,)
|
|   __basicsize__ = 864
|
|   __dictoffset__ = 264
|
|   __flags__ = 2148291584
|
|   __itemsize__ = 40
|
|   __mro__ = (<class 'type'>, <class 'object'>)
|
|   __weakrefoffset__ = 368
```

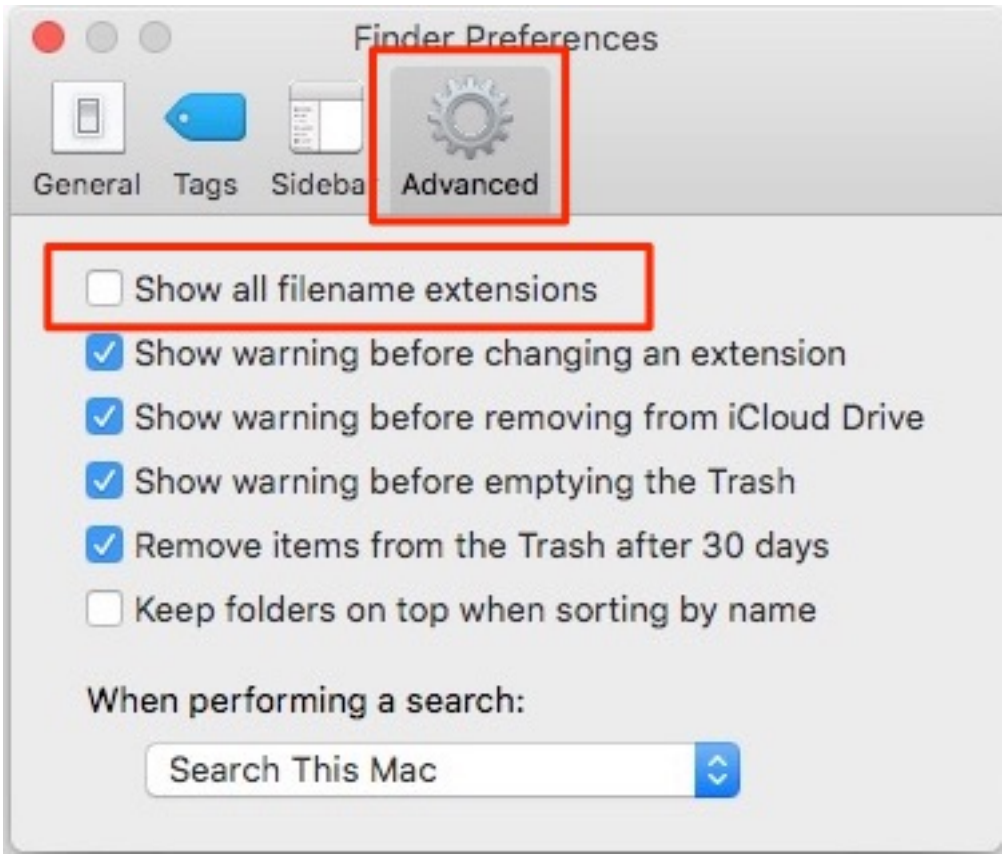
```
None
```

`script1.trtl.txt` vs. `script1.trtl`

In case you are not working on the command-line only, you might want to configure your File explorer/Finder as in the following.



`script1.trtl.txt` vs. `script1.trtl`



Text Editors

There are many text editors and even more for programming. Many of you will use Visual Studio Code (https://code.visualstudio.com/) in the next semester. If you want to, we can help you install it in the last session.

For now, you have for sure `TextEdit` (MacOS) and `Notepad` (Windows) installed on your computers. You can use these to inspect/edit plain text files.

What is this: `if __name__ == '__main__':`

That is the first content we are having a look at now... See

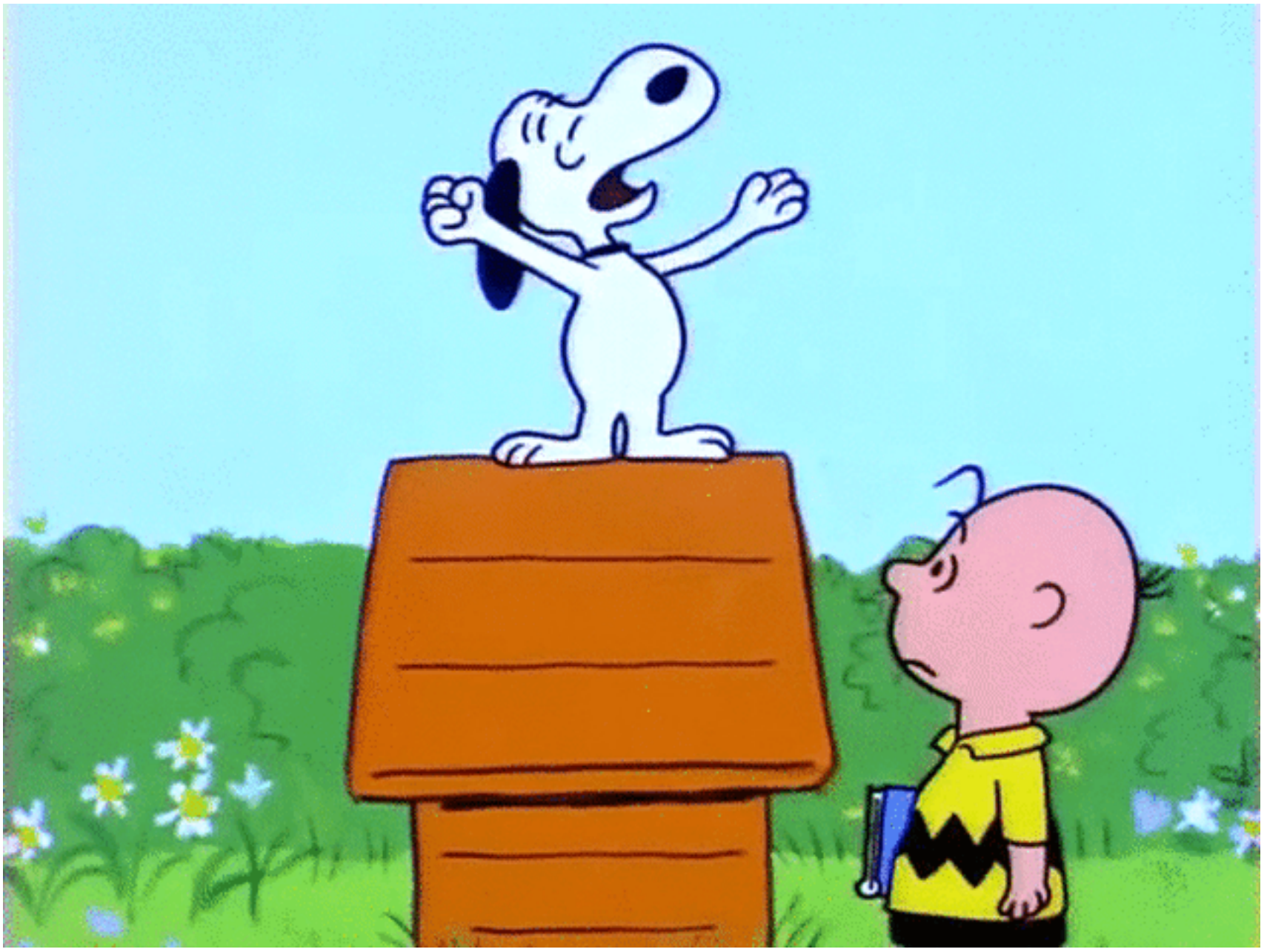
I still do not understand completely what the issue was with LearnIT yesterday.

However, I just reopened the hand-in and the group selection item (in case it is needed).

Can you please try to hand-in again. We just tried it with one of you and it should work now.

Please hand-in before 10am please.

Good morning!



IMPORTANT!!!

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

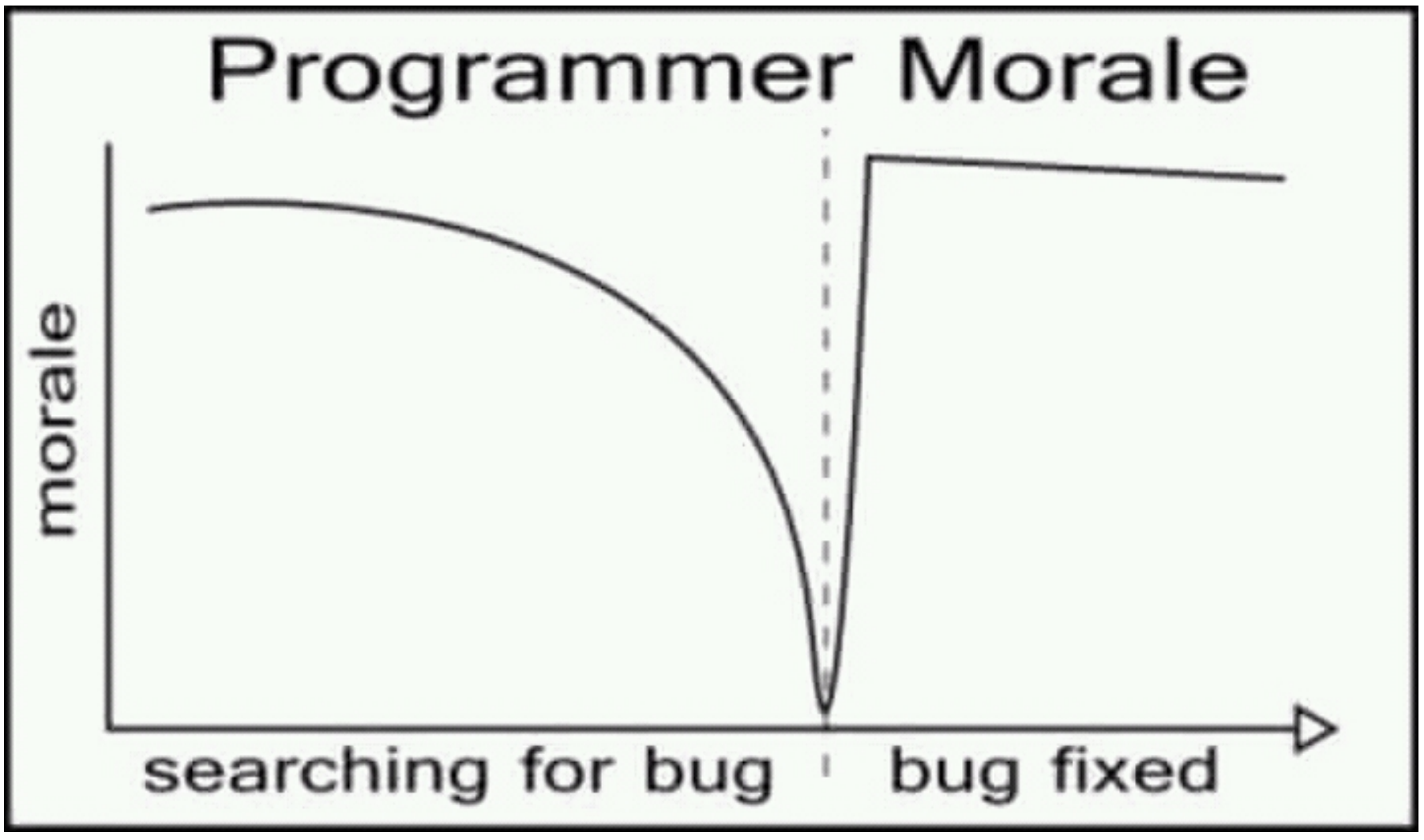
Otherwise you cannot enter the room and participate in it...

Assignment Feedback

Hand-in Technical Issue

We are sorry! We do not know what was going on.

Likely you experienced something like this during the assignment



Turtle Runner

Do you have an idea of why I am thinking this was one of the coolest exercises in the entire seminar?

In [ ]:

Turtle Geometry

In [ ]:

Loops

`for` loops and `while` loops

They can express the same. Remember Morten's slide!: This:

In [19]:

```
1
elephant
4
rats
```

is the same as

In [ ]:

But I would like to write a `for` loop but still need the index

You may want to use the `enumerate` function.

In [17]:

```
Help on class enumerate in module builtins:

class enumerate(object)
 |  enumerate(iterable, start=0)
 |
 |  Return an enumerate object.
 |
 |    iterable
 |      an object supporting iteration
 |
 |  The enumerate object yields pairs containing a count (from start
, which
 |  defaults to zero) and a value yielded by the iterable argument.
 |
 |  enumerate is useful for obtaining an indexed list:
 |      (0, seq[0]), (1, seq[1]), (2, seq[2]), ...
 |
 |  Methods defined here:
 |
```

In [27]:

```
0
1.0
rats
1
elephant
1.0
2
4.0
elephant
3
rats
4.0
```

The last basic data type: `tuple`

A list:

In [5]:

A tuple, is an immutable list

In [28]:

```
elephant
4
```

In [29]:

```
mouse
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-29-588241f7057d> in <module>
      4 my_list[1] = 'mouse'
      5 print(my_list[1])
----> 6 my_tuple[1] = 'mouse'
      7 print(my_tuple[2])

TypeError: 'tuple' object does not support item assignment
```

The `type` function revisited

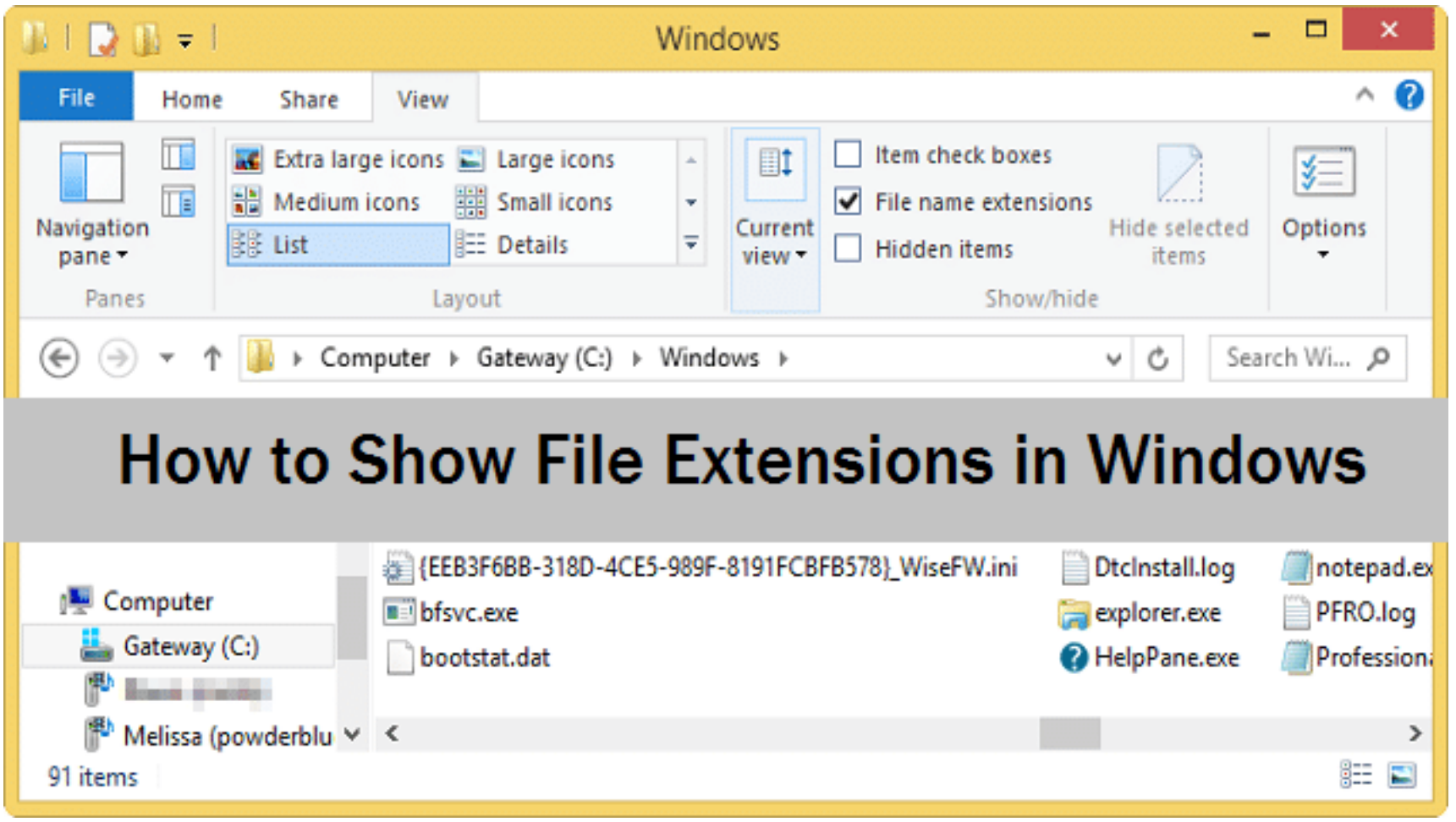
Now, that we started with object-oriented programming, the `type` function likely makes more sense. You give it an object and it tells you of which class (type) that object is an `instanceof`.

In [12]:

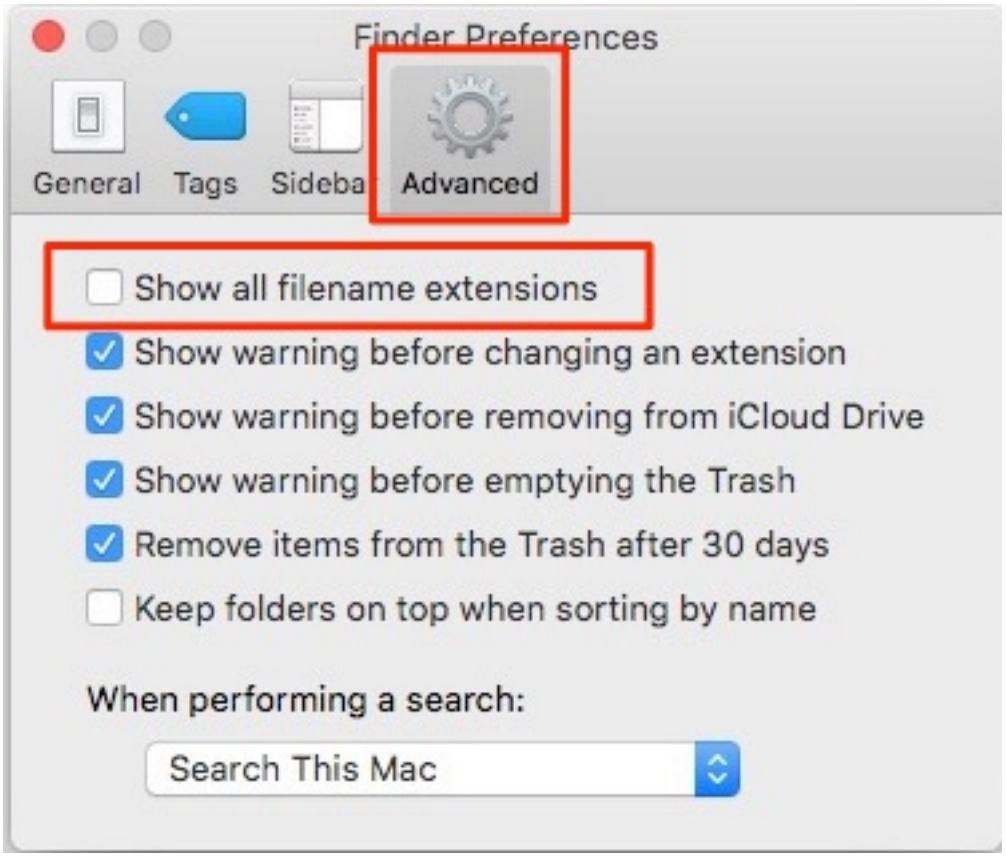
```
Help on class type in module builtins:

class type(object)
 |  type(object_or_name, bases, dict)
 |  type(object) -> the object's type
 |  type(name, bases, dict) -> a new type
 |
 |  Methods defined here:
 |
 |  __call__(self, /, *args, **kwargs)
 |      Call self as a function.
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __dir__(self, /)
 |      Specialized __dir__ implementation for types.
 |
 |  __getattribute__(self, name, /)
```

`script1.trtl.txt` vs. `script1.trtl`

In case you are not working on the command-line only, you might want to configure your File explorer/Finder as in the following.



`script1.trtl.txt` vs. `script1.trtl`



Text Editors

There are many text editors and even more for programming. Many of you will use Visual Studio Code (https://code.visualstudio.com/) in the next semester. If you want to, we can help you install it in the last session.

For now, you have for sure `TextEdit` (MacOS) and `Notepad` (Windows) installed on your computers. You can use these to inspect/edit plain text files.

What is this: `if __name__ == '__main__':`

That is the first content we are having a look at now... See