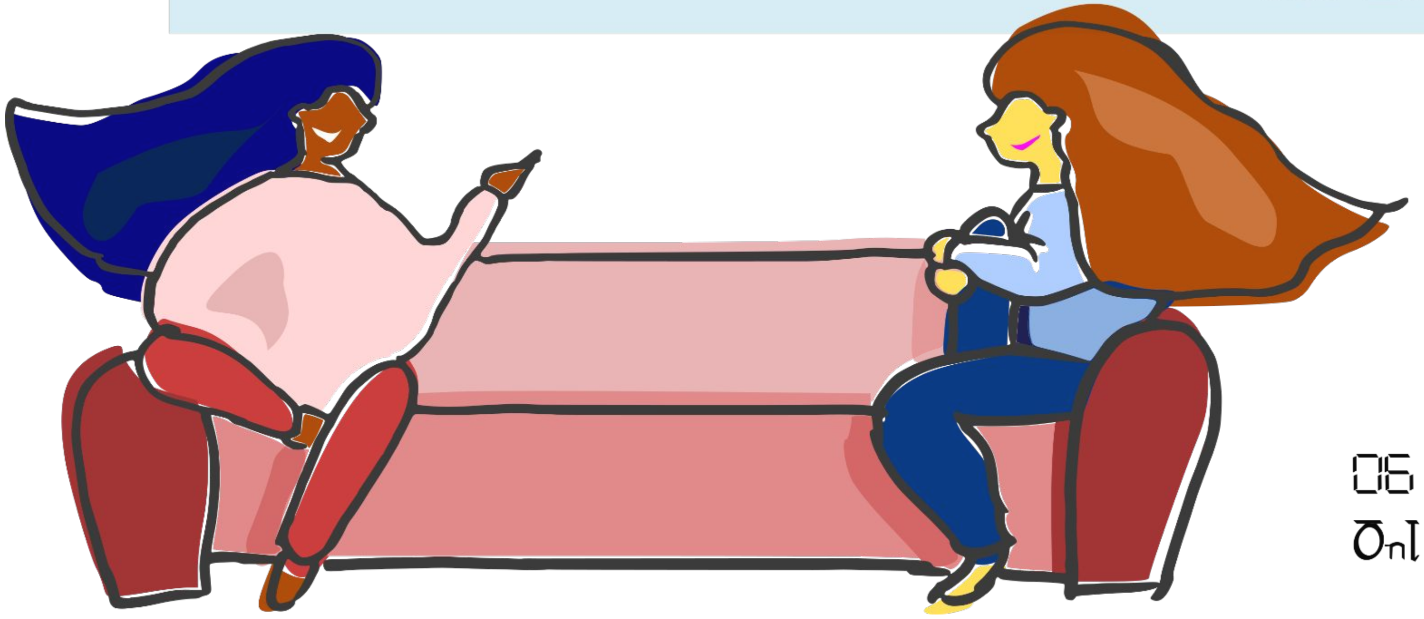
Let's talk about decorators

led by Maria Teleńczuk
Maud Levy



06 July 2021
Online, 7:30 pm

Introduction

Find us:

meetup.com/pyladiesparis

slackin.pyladies.com > #city-paris

https://twitter.com/pyladiesparis

https://github.com/PyLadiesParis

Let’s get to know each other!

Tell us about yourself,

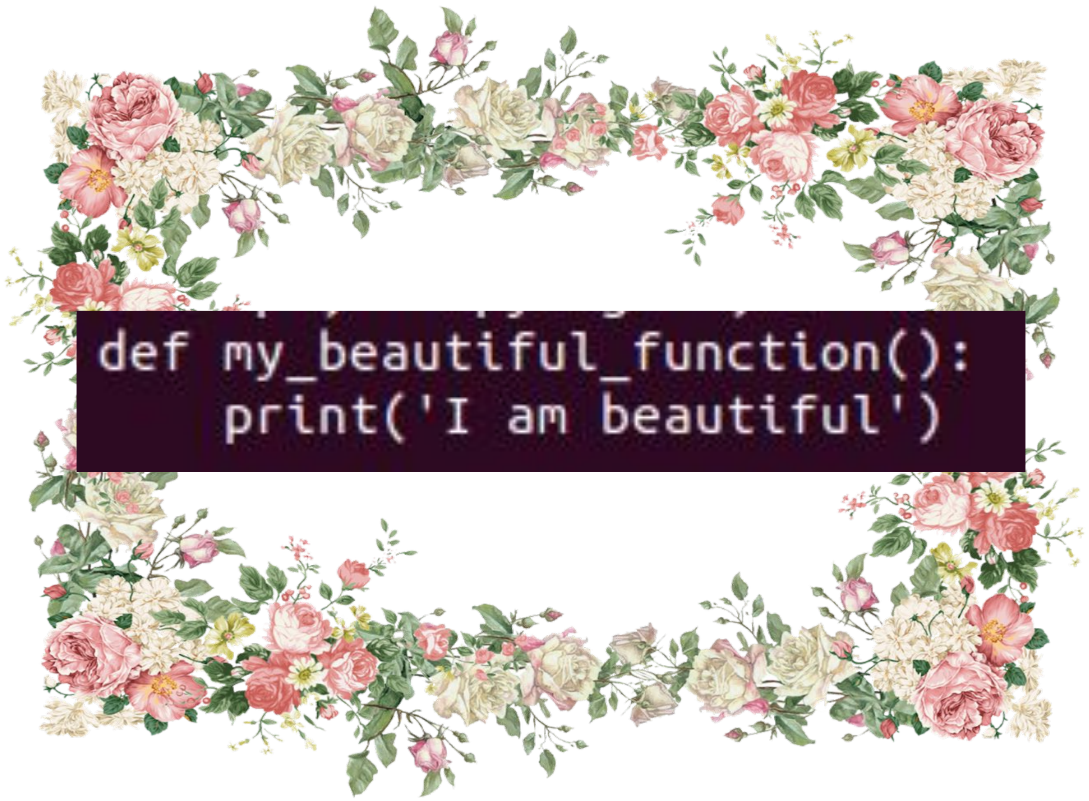
and..

About your relationship with Python

Decorators, what are they?

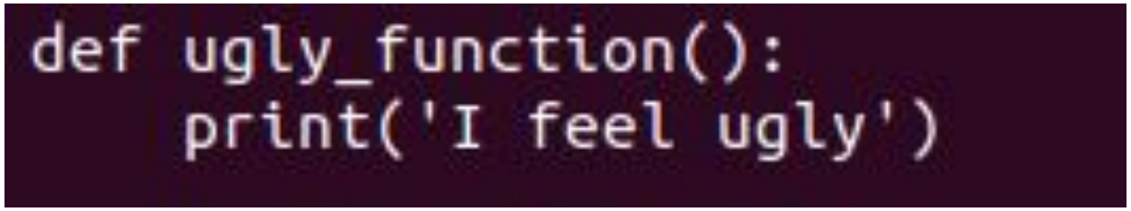
Decorators, what are they?

Decorate a function or a class



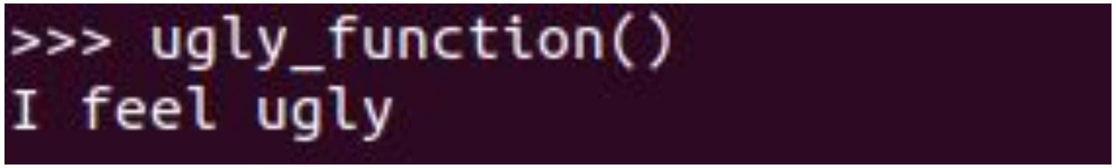
```
def my_beautiful_function():
    print('I am beautiful')
```

Function in Python? What is it?



```
def ugly_function():
    print('I feel ugly')
```

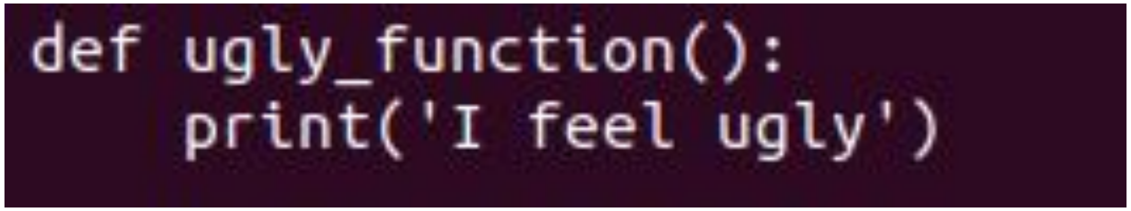
Output:



```
>>> ugly_function()
I feel ugly
```

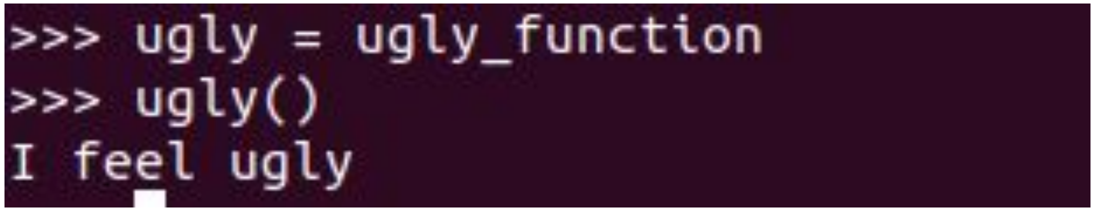
Function in Python? What is it?

Functions in Python are objects



```
def ugly_function():
    print('I feel ugly')
```

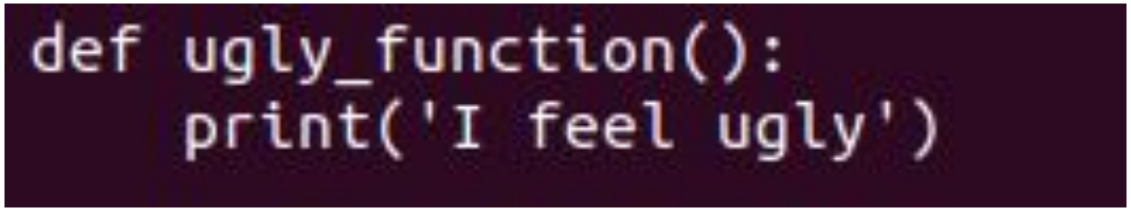
1. Can store it in a variable and call it later:



```
>>> ugly = ugly_function
>>> ugly()
I feel ugly
```

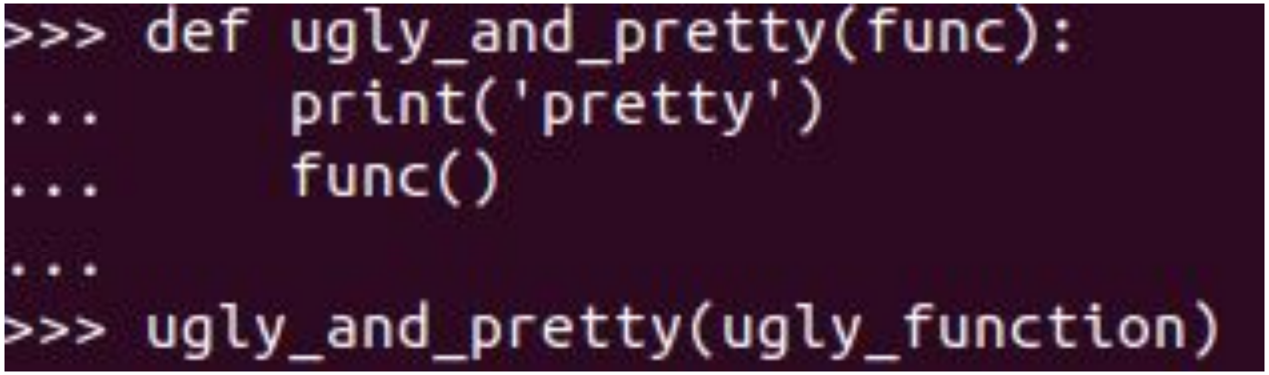
Function in Python? What is it?

Functions in Python are objects



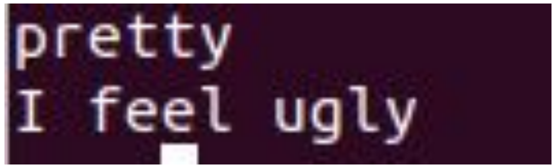
```
def ugly_function():
    print('I feel ugly')
```

2. Can pass it as a variable to another function



```
>>> def ugly_and_pretty(func):
...     print('pretty')
...     func()
...
>>> ugly_and_pretty(ugly_function)
```

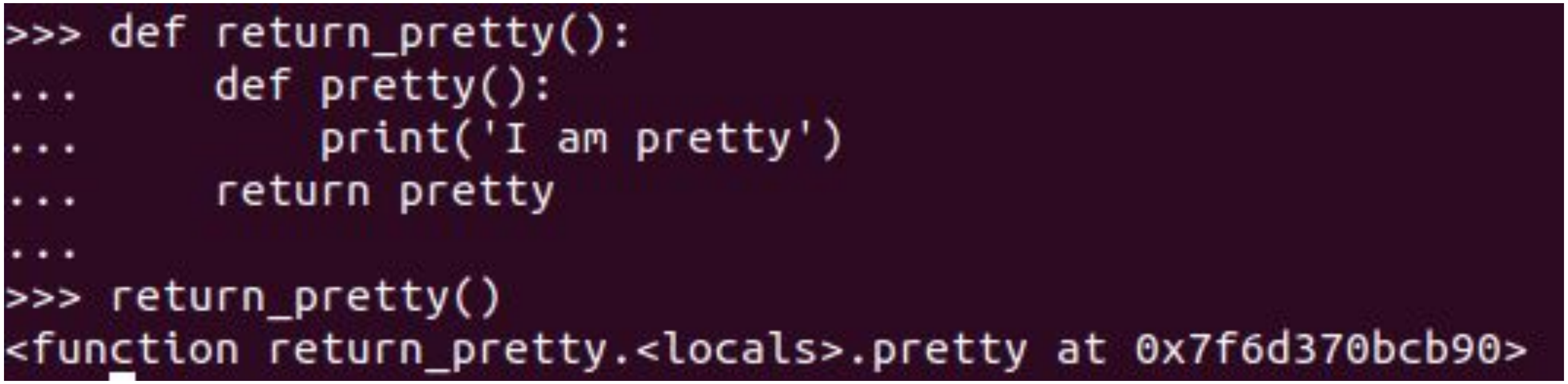
output:



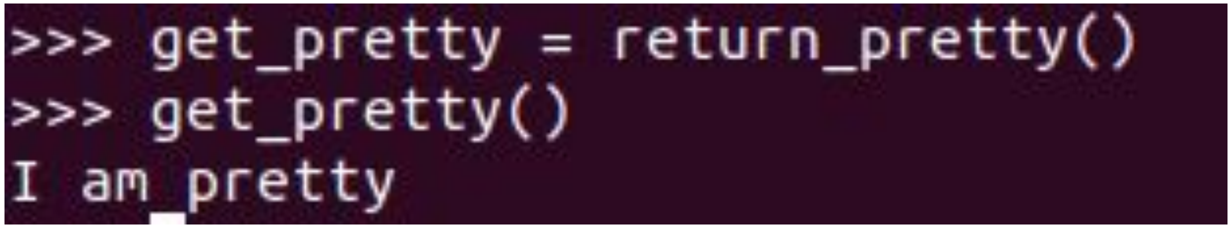
```
pretty
I feel ugly
```

Function in Python? What is it?

3. Can be returned as an output



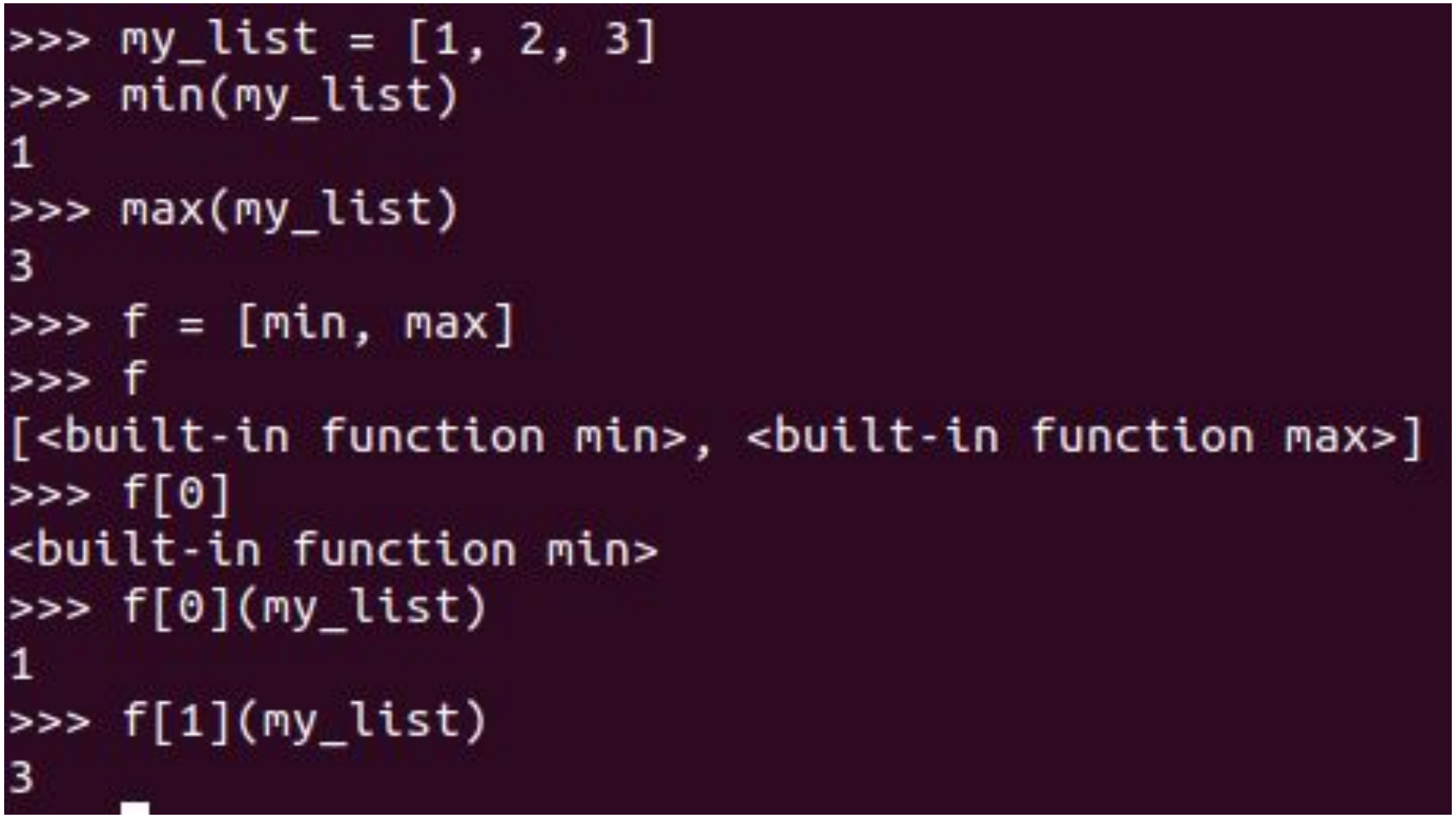
```
>>> def return_pretty():
...     def pretty():
...         print('I am pretty')
...     return pretty
...
>>> return_pretty()
<function return_pretty.<locals>.pretty at 0x7f6d370bcb90>
```



```
>>> get_pretty = return_pretty()
>>> get_pretty()
I am pretty
```

Function in Python? What is it?

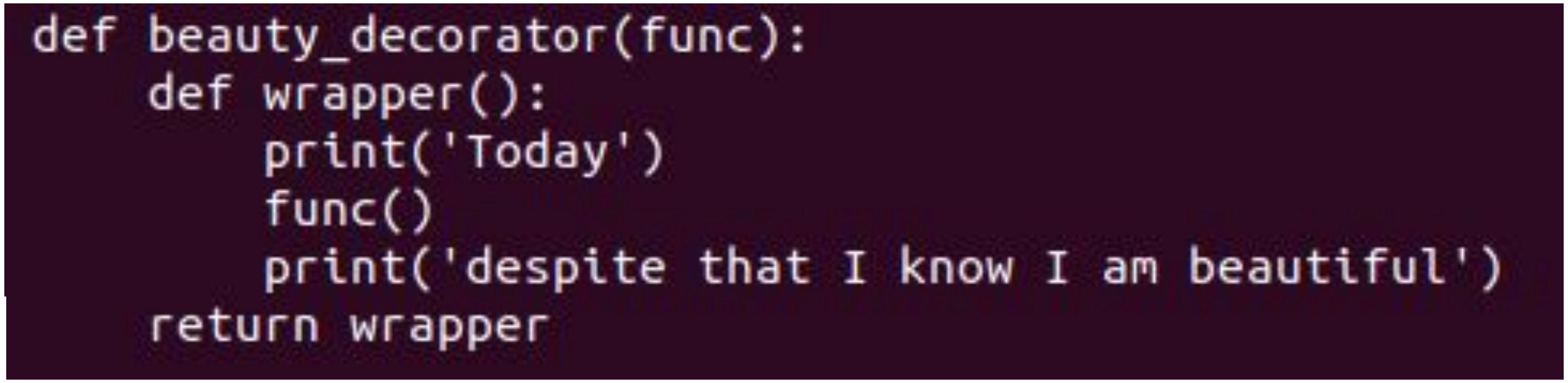
4. You can store them in data structures, eg:



```
>>> my_list = [1, 2, 3]
>>> min(my_list)
1
>>> max(my_list)
3
>>> f = [min, max]
>>> f
[<built-in function min>, <built-in function max>]
>>> f[0]
<built-in function min>
>>> f[0](my_list)
1
>>> f[1](my_list)
3
```

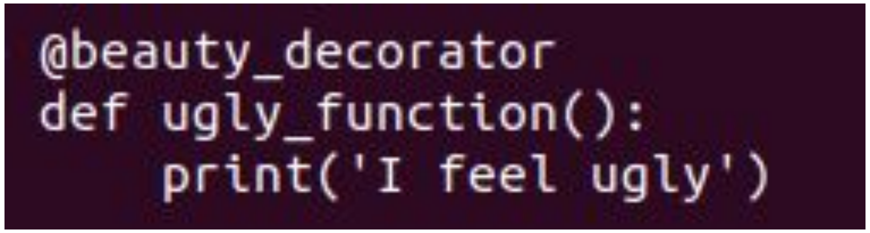
Decorators, what are they?

Define decorator:



```
def beauty_decorator(func):
    def wrapper():
        print('Today')
        func()
        print('despite that I know I am beautiful')
    return wrapper
```

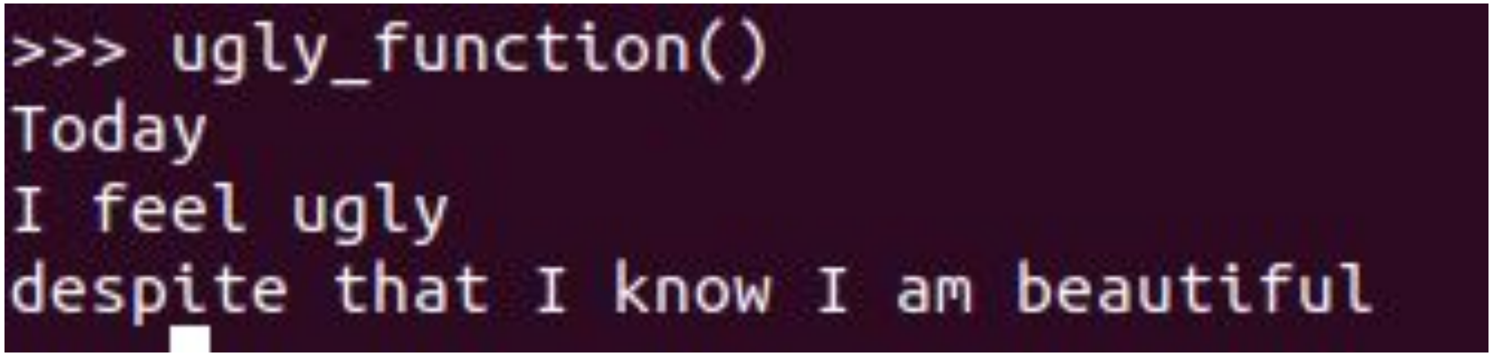
Define function + decorate it:



```
@beauty_decorator
def ugly_function():
    print('I feel ugly')
```

Decorators, what are they?

Output:



```
>>> ugly_function()
Today
I feel ugly
despite that I know I am beautiful
```

Decorators, some characteristics:

- Allow for modification of the behaviour of the function
- Extend behaviour of the function
- Used in tests (eg pytest @parametrize)
- Possible chaining of decorators

Commonly used decorators

Within the custom class:

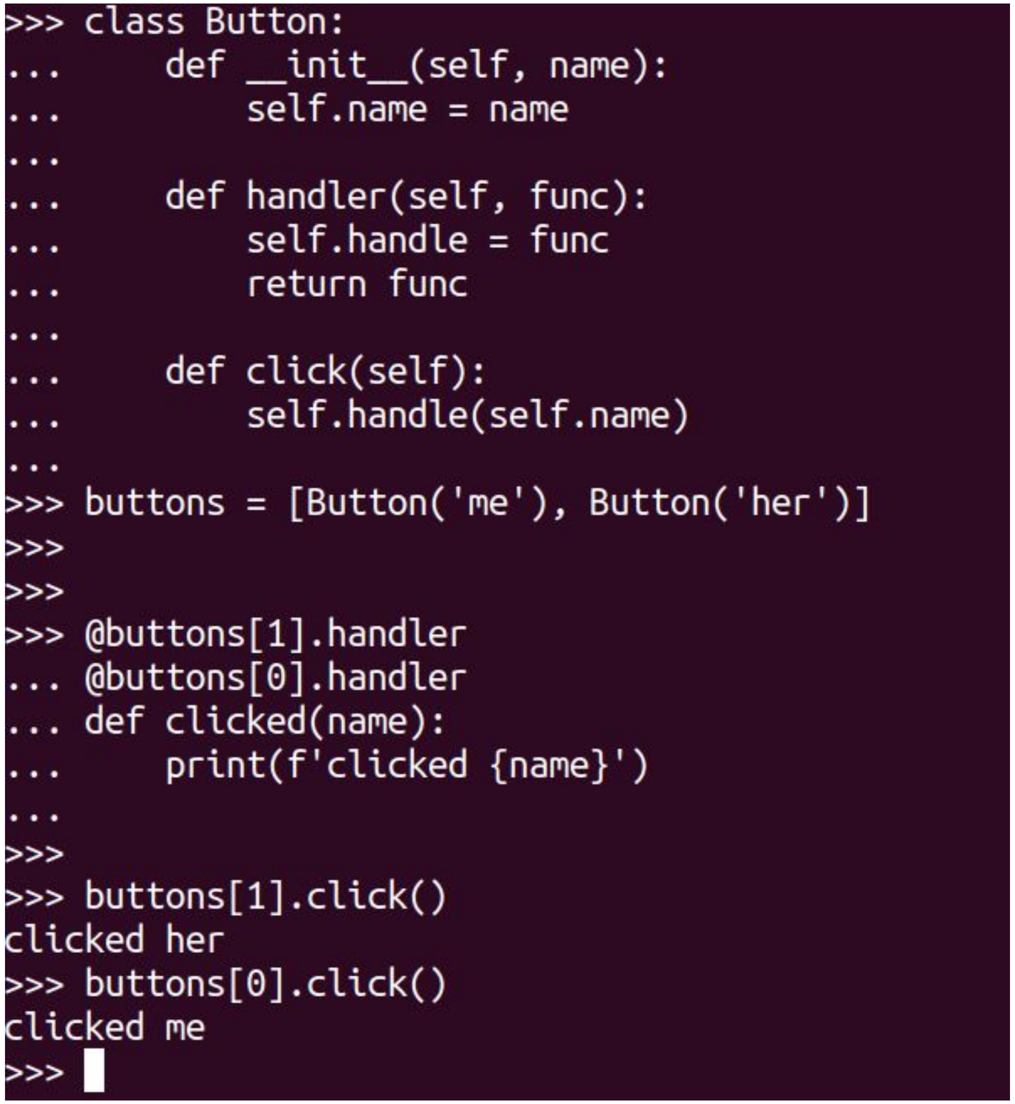
- @classmethod : can only access class attributes, but not instance attributes; can be used to get an object of the class
  - pass a class as a first argument

- @staticmethod : performs operation independent of the class
  - don’t pass self nor class

- @property : creates a property from an instance method (can now access some functions as they were attributes)
  - pass self

Extracted from Python 3.9 presentation

Decorators can now consist of any valid expression



```
>>> class Button:
...     def __init__(self, name):
...         self.name = name
...
...     def handler(self, func):
...         self.handle = func
...         return func
...
...     def click(self):
...         self.handle(self.name)
...
>>> buttons = [Button('me'), Button('her')]
>>>
>>>
>>> @buttons[1].handler
... @buttons[0].handler
... def clicked(name):
...     print(f'clicked {name}')
...
>>>
>>> buttons[1].click()
clicked her
>>> buttons[0].click()
clicked me
>>> 
```

What do you think about decorators?

Future of Pyladies Paris

1. What kind of events mostly interest you?

2. Would you like to be a speaker of a talk in the future?

- Tell us about yourself: https://kutt.it/ppspeaking
- Relaxed discussions suggestions: https://kutt.it/ppdiscussions

3. What can we improve?

   Share your feedback: https://kutt.it/ppfeedback

spoiler for the next event:
open source

Thanks!!!