Classes as namespaces

Tomaž Šolc
tomaz.solc@tablix.org

Python Meetup Ljubljana
13 June 2019

The usual

```
class Config:
  def __init__(self):
    self.color = “green”

def main():
  config = Config()
  print(“my favorite color is”,
        config.color)
```

No need for an instance?

```
class Config:
  color = “green”

def main():
  print(“my favorite color is”,
        Config.color)
```

If statement under class

```
from sys import platform

class Config:
  if platform.startswith(“darwin”):
    color = “black”
  else:
    color = “green”

def main():
  print(“my favorite color is”,
        Config.color)
```

Arbitrary code under class?!

```
from random import randint

class Config:
  for n in range(10):
    if randint(1,6) == 6:
      color = “black”
      break
  else:
    color = “red”

...
```

Interpreting a class block

- Create a new execution frame
  - Like for a function call
    but you can’t call return from it :(
  - New empty locals(), existing globals()
- Execute code block
- Create a new class object
  - Copy contents of locals() to __dict__ of new object
    ignoring inheritance here for simplicity
- Discard execution frame, locals()

Please use responsibly

Thanks

Errata:

In the discussion that followed my talk, it was pointed out that variable scoping works differently in a class block than in a function. Hence some code might not work as expected. One common example are list comprehensions, as explained here:

https://stackoverflow.com/a/13913933