esade

Advanced Programming in Python

LECTURE 2

Do Good. Do Better.

esade

Previously On...

Previously on...

Inmutable Types:

Boolean:

```
True, False, and, or, not...
```

Numeric:

```
n_int = 1
n_float = 1.1
n_complex = 1 + 1j
```

Strings:

```
str = "Hello"
str = 'Hello'
```

Mutable Types:

Lists (0 based indexing):

```
lst = [1,2,3,4]
lst.append(), lst.clear()
```

Dictionary:

```
dct = {"apple": 1, "orange": 2}
dct.update({"pineapple":3})
```

Python Org, April 2022, https://docs.python.org/3/library/stdtypes.html (Accessed: 20-04-2022)

Previously on...

Inmutable Types:

Tuples:

```
tpl = (1,2,3,4)
```

Ranges:

```
rng = range(0,10) -> [0..10)
```

Frozenset:

```
st = frozenset([1,2,3,4])
```

Mutable Types:

Set:

```
st = set([1,2,4,4]) -> (1,2,4)
st.add(5), st.remove(4)
```

List as stack:

```
lst.pop(), lst.append(1)
```

Python Org, April 2022, https://docs.python.org/3/library/stdtypes.html (Accessed: 20-04-2022)

Previously on...

Bare minimum requirements of a program?

- Import libraries:
```
import sys
```
- Functions:
```
def function(param1):
```
- Main entry point :
```
if __name__ == "__main__":
```

Previously on...

Branching

```
def some_function(a):
    if a == 1:
        s = "Hello"
    elif a == 2:
        s = "Bye"
    else:
        s = "End"
```

Looping

```
while condition:
    pass

for elem in sequence:
    pass
```

Pythonic way

```
sequence = [7,8,9,10,11,12]
elem = [n for n in sequence]
```

esade

Introduction

Specs, OOP and more...

Outline

Previously on...

Introduction

Specifications

Development Setup

Object Oriented Programming.

Architecture.

Live Demo.

esade

Specifications

Do Good. Do Better.

Specifications

Conceptual Specifications

Scope of the project

Functionality

Constraints

Cost:

- Personel
- Infrastructure

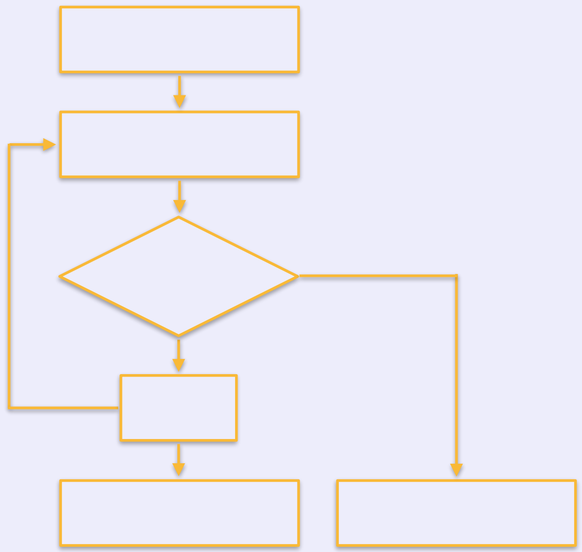
Technical Specifications

Code specification

- Requirements
- Min version
- ...

Fault tolerance

Reusability



Example

esade

Development Setup

Do Good. Do Better.

Development Setup

IDE

- Visual Studio Code
- Extensions:
  - Python
  - GitLens

Minimum Requirements

- Python 3.8
- Virtual environment

```
python3 -m virtualenv -p python3 ./env

source env/bin/activate
```

- Install libraries

```
python3 -m pip install <package>
```

esade

Object Oriented Programming

With Demos...

Object Oriented Programming

Variables

Encapsulation

Class

Namespaces

Methods

Objects

OOP: Classes

```
class Fruit:
    def __init__(self, size: int, name: str) -> None:
        print("init {}".format(id(self)))
        self.size = size
        self.name = name

if __name__ == "__main__":
    c1 = Fruit(10, "orange")
    c2 = Fruit(5, 'apple')
```

OOP: Methods

```
class Fruit:
    def __init__(self, size: int, name: str) -> None:
        self.size = size
        self.name = name

    def set_size(self, size: int = 10) -> None:
        self.size = size

if __name__ == "__main__":
    c1 = Fruit(10, "orange")
    c1.set_size(15)
```

OOP: Objects

```
class Fruit:
    def __init__(self, size: int, name: str) -> None:
        self.size = size
        self.name = name

    def set_size(self, size: int = 10) -> None:
        self.size = size

if __name__ == "__main__":
    c1 = Fruit(10, "orange")
    c1.name = "pineapple"
    c1.set_size(15)
```

OOP: Namespace

- Namespaces lets us uniquely identify each variable and object within our program.

- Namespaces provide the mapping between the names (functions and objects) to the actual instance/reference.

- As an example, when we instantiate a class, we create a new namespace for that class.

OOP: Encapsulation or scope

```
class Fruit:
    color = 'orange' # no mutable
    attributes = [] # mutable -> this is a problem!
    def __init__(self, size: int, name: str) -> None:
        ...
        self.name = name
        self.__other = [] # private variable

if __name__ == "__main__":
    c1 = Fruit(10, "orange")
    c2 = Fruit(5, 'apple')
    c2.color = 'green'
    c2.attributes.append('something')
```

OOP: Inheritance

```
class Computer:
    def __init__(self, ram: int = 8, cpu_core: int=8) -> None:
        self.ram = ram
        self.cpu_core = cpu_core

class Laptop(Computer):
    def __init__(self, screen: int=15, usb: int=2):
        self.screen = screen
        self.usb = usb

if __name__ == "__main__":
    l = Laptop()
    l.ram = 16
```

OOP: lambda functions

```
def sort_even(number):
    return (number%2)

ll = [1,2,3,4,5,6]
ll.sort(key=sort_even)

# [2, 4, 6, 1, 3, 5]
```

Anonymous Function

```
ll = [1,2,3,4,5,6]
ll.sort(key=lambda val: (val%2))

# [2, 4, 6, 1, 3, 5]
```

OOP: Dataclasses

```
from dataclasses import dataclass

@dataclass
class person:
    name: str
    surname: str
    age: int
    height: int

if __name__ == "__main__":
    p1 = person("John", "Doe", 30, 180)
```

esade

Architecture

Technical definition & Execution model

Do Good. Do Better.

LIVE DEMO

Bibliography

Python Tutorial: Classes, April 2022, https://docs.python.org/3/tutorial/classes.html (Accessed: 20-04-2022)

Python Docs: Dataclasses, April 2022, https://docs.python.org/3/library/dataclasses.html (Accessed: 20-04-2022)

Python Docs: Lambdas, April 2022, https://docs.python.org/3/reference/expressions.html?highlight=lambda (Accessed: 20-04-2022)

esade

Do Good. Do Better.