Лекция 11

Язык программирования Python.

Хайрулин Сергей Сергеевич

email: s.khairulin@g.nsu.ru, s.khayrulin@gmail.com

Ссылка на материалы

План

- Лекции/практические занятия
  - Тест
- Дифференцированный зачет в конце семестра
  - Защита задания

Литература

Начальный уровень

- Mark Pilgrim. Dive into Python - http://www.diveintopython.net/
- Марк Лутц. Изучаем Python, 4-е издание // Символ-Плюс 2011.
- ...

Стандарт/Документация

- PEP-8 - https://www.python.org/dev/peps/pep-0008/
- https://www.python.org/
- https://github.com/python/cpython

Экспертный уровень

- Лучано Рамальо: Python. К вершинам мастерства
- Mitchell L. Model. Bioinformatics Programming Using Python // O'Reilly 2010.

Версии Python

- Python 2 вышел 2010 году последняя версия 2.7.16 - исправлялись только баги(ошибки) с января 2020 года поддержка прекращена.
- Python 3 в появился в 2008, является актуальной версией языка. Текущая стабильная версия 3.9, в разработке 3.10
  - Python 3 не гарантирует совместимости кода с Python 2

Summary

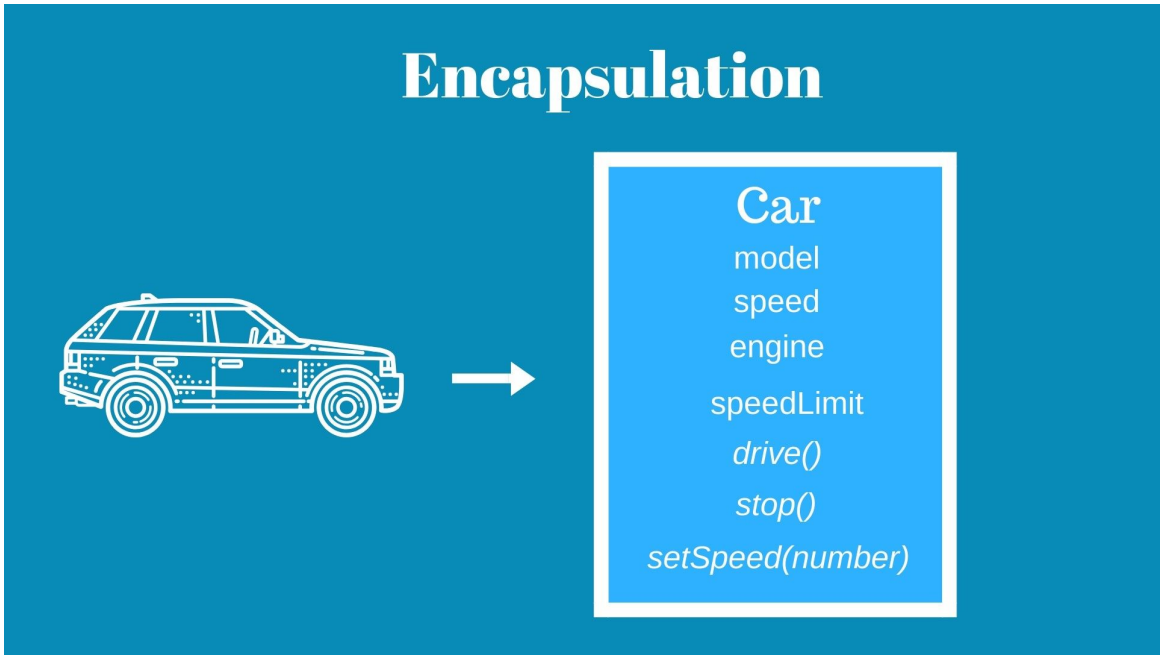
- Инкапсуляция/сокрытие
  - реализации в Python.
- Обработка исключительных ситуаций.
  - Конструкция: **try … catch …**
  - Пользовательские классы исключений

ООП принципы

- Наследование
  - Возможность создание новых типов данных базирующихся на других, ранее определенных
- Полиморфизм
  - Возможность переопределения поведения базовых свойств объекта (свойств унаследованных от объектов предков)
- Инкапсуляция
  - Возможность скрывать реализацию тех или иных свойств объекта от конечного пользователя

Инкапсуляция

Свойства языка, позволяющее скрыть реализацию или данные от конечного пользователя.



Инкапсуляция

incapsultion.py > ...

```python
import math


class Circle:
    def __init__(self, radius, center_x=0, center_y=0):
        self.radius = radius
        self.center_x = center_x
        self.center_y = center_y

    def area(self):
        return math.pi * self.radius ** 2


if __name__ == '__main__':
    c = Circle(10)
    print(c.area())

```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[13:40:40] serg :: serg-pc  → ~/tmp/oop»
python3 incapsultion.py
314.1592653589793
[13:43:54] serg :: serg-pc  → ~/tmp/oop»
```

Инкапсуляция

```
...
# Где-то в программе поменяли радиус изначальной окружность
# при этом это сильно влияет на результаты работы
# так как мы не предполагаем, что это свойстро должно меняться.
c.radius = 20
# Кроме того, безконтрольный доступ к полям, может привести к
# аварийным ситуациям.
c.radius = "incorrect radius" # После этого метод area() перестанет работать
c.radius = -1 # После этого метод area() будет возвращать невероятные результаты
...
```

Инкапсуляция

Реализации в Python.
Во многих языках программирования.
Ограничение доступа к полям и методам, организовывается с помощью ключевых слов (public/private). В Python для этого - существует соглашение об именовании.

incapsultion.py > ...

```python
import math


class Circle:
    def __init__(self, radius, center_x=0, center_y=0):
        self.__radius = radius
        self.__center_x = center_x
        self.__center_y = center_y

    def area(self):
        return math.pi * self.__radius ** 2


if __name__ == '__main__':
    c = Circle(10)
    print(c.area())
    print(c.__radius)

```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[13:54:22] serg :: serg-pc  ➜  ~/tmp/oop»
python3 incapsultion.py
314.1592653589793
Traceback (most recent call last):
  File "incapsultion.py", line 17, in <module>
    print(c.__radius)
AttributeError: 'Circle' object has no attribute '__radius'
```

Инкапсуляция

```python
class Circle:
    def __init__(self, radius, center_x=0, center_y=0):
        self.__radius = radius
        self.__center_x = center_x
        self.__center_y = center_y

    def area(self):
        return math.pi * self.__radius ** 2

    def on_circle(self, fig):
        for vertx_x, vertex_y in fig:
            if self.__on(vertx_x, vertex_y) is False:
                return False
        return True

    def __on(self, point_x, point_y):
        if self.__dist(point_x, point_y) == self.__radius:
            return True
        return False

    def __dist(self, point_x, point_y):
        return math.sqrt((self.__center_x - point_x)**2 + (self.__center_y - point_y)**2)
```

Обработка аварийных/исключительных ситуаций.

```
In [1]: 1 / 0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-bc757c3fda29> in <module>
----> 1 1 / 0

ZeroDivisionError: division by zero
```

```
In [2]: 1 + "abc"
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-87be4b1bc3f8> in <module>
----> 1 1 + "abc"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Конструкция: try … catch …

exception.py > ...

```
1  if __name__ == '__main__':
2      try:
3          div = int(input("Input num: "))
4          print(1/div)
5      except ZeroDivisionError as err:
6          print(f"You're trying to dievice 1 by zero {err}")
7
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[14:45:09] serg :: serg-pc  ➜  ~/tmp/oop»
python3 exception.py
Input num: 10
0.1
[14:45:16] serg :: serg-pc  ➜  ~/tmp/oop»
python3 exception.py
Input num: 0
You're trying to dievice 1 by zero division by zero
[14:45:46] serg :: serg-pc  ➜  ~/tmp/oop»
```

Конструкция: try … catch …

exception.py > ...

```python
if __name__ == '__main__':
    try:
        div = int(input("Input num: "))
        print(1/div)
    except ZeroDivisionError as err:
        print(f"You're trying to dievice 1 by zero {err}")
    except ValueError as err:
        print(f"Wrong value err: {err}")

```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[14:55:18] serg :: serg-pc  ➜ ~/tmp/oop»
python3 exception.py
Input num: dsds
Wrong value err: invalid literal for int() with base 10: 'dsds'
```

Конструкция: try … catch …

```python
if __name__ == '__main__':
    try:
        div = int(input("Input num: "))
        print(1/div)
    except ZeroDivisionError as err:
        print(f"You're trying to dievice 1 by zero {err}")
    except ValueError as err:
        print(f"Wrong value err: {err}")
    except Exception:
        print(f"Some bad happend: {err}")
    except:  # <=> except BaseException
        print(f"Some REALY bad happend: {err}")
```

Пользовательские классы исключений

BaseException - системные ошибки.

Exception - Все встроенные исключения, не связанные с системными ошибками, являются производными от этого класса. Все пользовательские исключения также должны быть производными от этого класса.

Полный список встроенных исключений можно найти здесь.

Пользовательские классы исключений

С помощью ключевого слова raise - можно возбуждать исключение, при этом программа останавливает выполнение и управление передается подходящему блоку catch - тип исключение соответствует типу обрабатываемого. Если подходящего блок не найден, то программа прекращает выполнение и останавливается, при этом в стандартный поток вывода, выводиться информация об ошибке.

```
class MyException(Exception):
    def __str__(self):
        return "HaHa you're on the wrong way"


if __name__ == '__main__':
    try:
        raise MyException()
    except MyException as e:
        print(e)
```

Пользовательские классы исключений

```python
from math import sqrt


class SQRTError(Exception):
    def __str__(self):
        return "Check you're input case we're not working with complex number"


def f(y):
    if y == 0:
        raise ZeroDivisionError("Y is equal to 0")
    return 1/y


def my_sqrt(n):
    if n < 0:
        raise SQRTError()
    return sqrt(n)


if __name__ == '__main__':
    try:
        f(2)
        my_sqrt(-1)
    except ZeroDivisionError as e:
        print(e)
    except SQRTError as e:
        print(e)

```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

[15:09:26] serg :: serg-pc  → ~/tmp/oop»
python3 exception.py
Y is equal to 0
[15:09:27] serg :: serg-pc  → ~/tmp/oop»
python3 exception.py
Check you're input case we're not working with complex number
```

Спасибо за внимание. Вопросы?