ligarobotov.ru

14-15 занятие

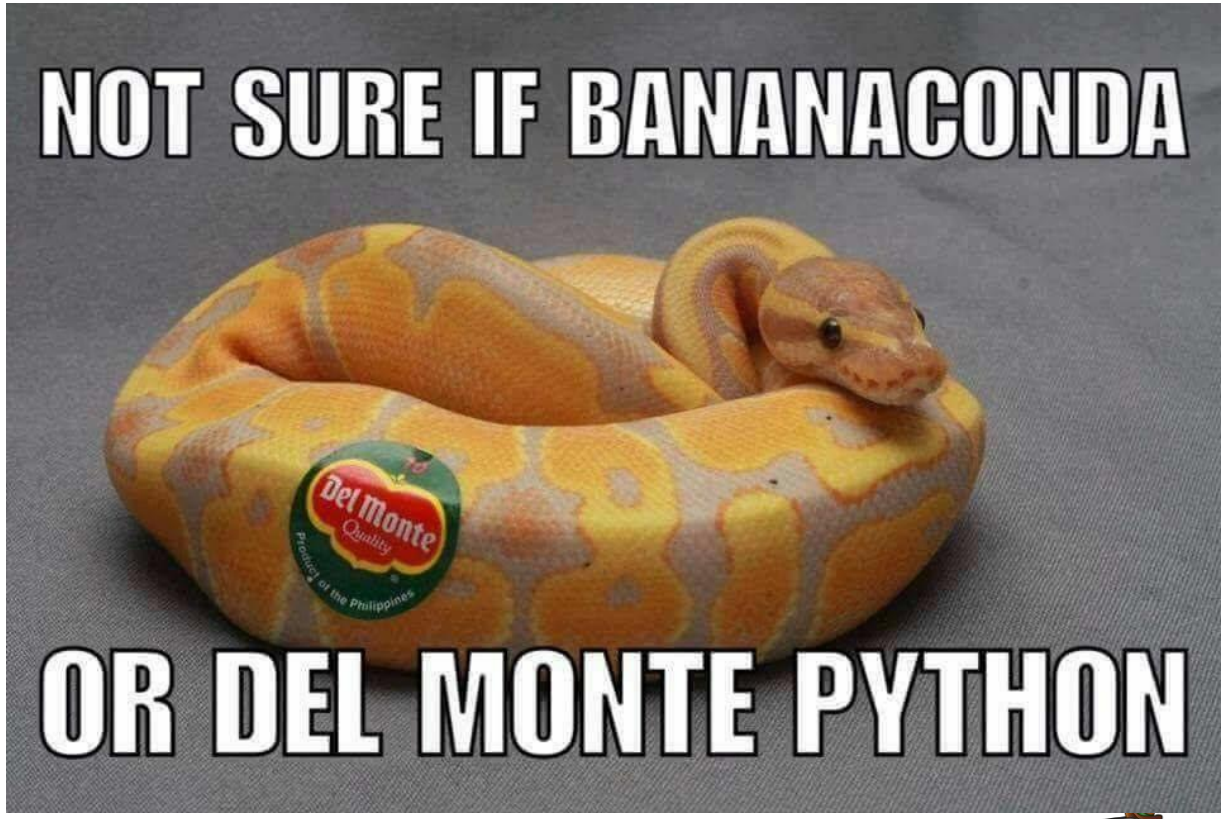
Федеральная сеть
секций робототехники
«Лига Роботов»
ligarobotov@gmail.com
89513839876

 Красноярск 2022

План занятия

1. Генераторы
2. Итераторы
3. Декораторы

Рубрика мемы



Генераторы

Что такое генератор и с чем его есть.

Генератор — это объект, который сразу при создании не вычисляет значения всех своих элементов.

Он хранит в памяти только последний вычисленный элемент, правило перехода к следующему и условие, при котором выполнение прерывается.

Генераторы

Вычисление следующего значения происходит лишь при выполнении метода next(). Предыдущее значение при этом теряется.

Генераторы

```
>>> a = (i**2 for i in range(1,5))
>>> a
<generator object <genexpr> at 0x0000023A7524D6D0>
>>> next(a)
1
>>> next(a)
4
>>> next(a)
9
>>> next(a)
16
>>> next(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

Генераторы

```
>>> next(gen)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

Генераторы

```
>>> next(gen)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

Генераторы

```
>>> a = (i**2 for i in range(1,5))
>>> for i in a:
...     print(i)
1
4
9
16
```

Генераторы

```
>>> a = (i**2 for i in range(1,5))
>>> for i in a:
...     print(i)
1
4
9
16
```

Генераторы

```
>>> def f_gen(m):
...     s = 1
...     for n in range(1,m):
...         yield n**2 + s
...         s += 1
...
>>> a = f_gen(5)
>>> a
<generator object f_gen at 0x0000023EE468D6D0>
>>> for i in a:
...     print(i)
...
2
6
12
20
>>>
```

Итераторы

```
>>> num_list = [1, 2, 3, 4, 5]
>>> for i in num_list:
    print(i)
1
2
3
4
5
```

Итераторы

```
>>> num_list = [1, 2, 3, 4, 5]
>>> for i in num_list:
    print(i)
1
2
3
4
5
```

Итераторы

```
>>> itr = iter(num_list)
>>> print(next(itr))
1
>>> print(next(itr))
2
>>> print(next(itr))
3
>>> print(next(itr))
4
>>> print(next(itr))
5
>>> print(next(itr))
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    print(next(itr))
StopIteration
```

Декораторы

```
def a_function():
    """Обычная функция"""
    return "1+1"

if __name__ == "__main__":
    value = a_function()
    print(value)
```

Декораторы

```
def another_function(func):
    """
    Функция которая принимает другую функцию.
    """

    def other_func():
        val = "Результат от %s это %s" % (func(),
            eval(func())
        )
        return val

    return other_func
```

Декораторы

```
def another_function(func):
    """
    Функция которая принимает другую функцию.
    """

    def other_func():
        val = "Результат от %s это %s" % (func(),
            eval(func())
        )
        return val

    return other_func
```

Декораторы

```
def another_function(func):
    """
    Функция которая принимает другую функцию.
    """

    def other_func():
        val = "Результат от %s это %s" % (func(),
            eval(func())
        )

        return val
    return other_func


def a_function():
    """Обычная функция"""
    return "1+1"

if __name__ == "__main__":
    value = a_function()
    print(value)
    decorator = another_function(a_function)
    print(decorator())
```

Декораторы

```
def another_function(func):
    """
    Функция которая принимает другую функцию.
    """

    def other_func():
        val = "Результат от %s это %s" % (func(),
            eval(func())
        )

        return val
    return other_func


def a_function():
    """Обычная функция"""
    return "1+1"

if __name__ == "__main__":
    value = a_function()
    print(value)
    decorator = another_function(a_function)
    print(decorator())
```

Самостоятельные задачи

Перебрать с помощью итератора список целых чисел [1, 3, 4, 5, 8]

Самостоятельные задачи

```
a = [1, 3, 4, 5, 8]
a_iter = a.__iter__()
print(a_iter)
print(type(a_iter))
try:
    print(a_iter.__next__())
    print(a_iter.__next__())
    print(a_iter.__next__())
    print(a_iter.__next__())
    print(a_iter.__next__())
    print(a_iter.__next__())
except StopIteration:
    print('Элементы закончились')
```

Самостоятельные задачи

Создать генератор чисел от 1 до 250 и перебрать его через цикл.

Самостоятельные задачи

```
def number():
    num = 0
    while True:
        yield num
        num += 1

for num in range(1,251):
    print(num)
```

Самостоятельные задачи

Реализовать простейший декоратор, который выводит любое сообщение. Создать обычную функцию, принимающую имя и вывести в консоль

Самостоятельные задачи

```
def clas(name):
    print(f'Открываем функцию {name}')
    name()
    print('Закрыли функцию')
    return clas
def name():
    print('Дима')
print(clas(name))
```