Рекурсия

Основы Python

План урока

- Рекурсия

Функции

ooo | код

```
def hello():
    print('hello')

hello()
```

ooo | вывод

```
>>> hello
```

Функции

ooo | код

```
def hello():
    print('hello')

def asterisks_hello():
    print('**start**')
    hello()
    print('***end***')


asterisks_hello()
```

ooo | вывод

```
>>> **start**
>>> hello
>>> ***end***
```

Функции

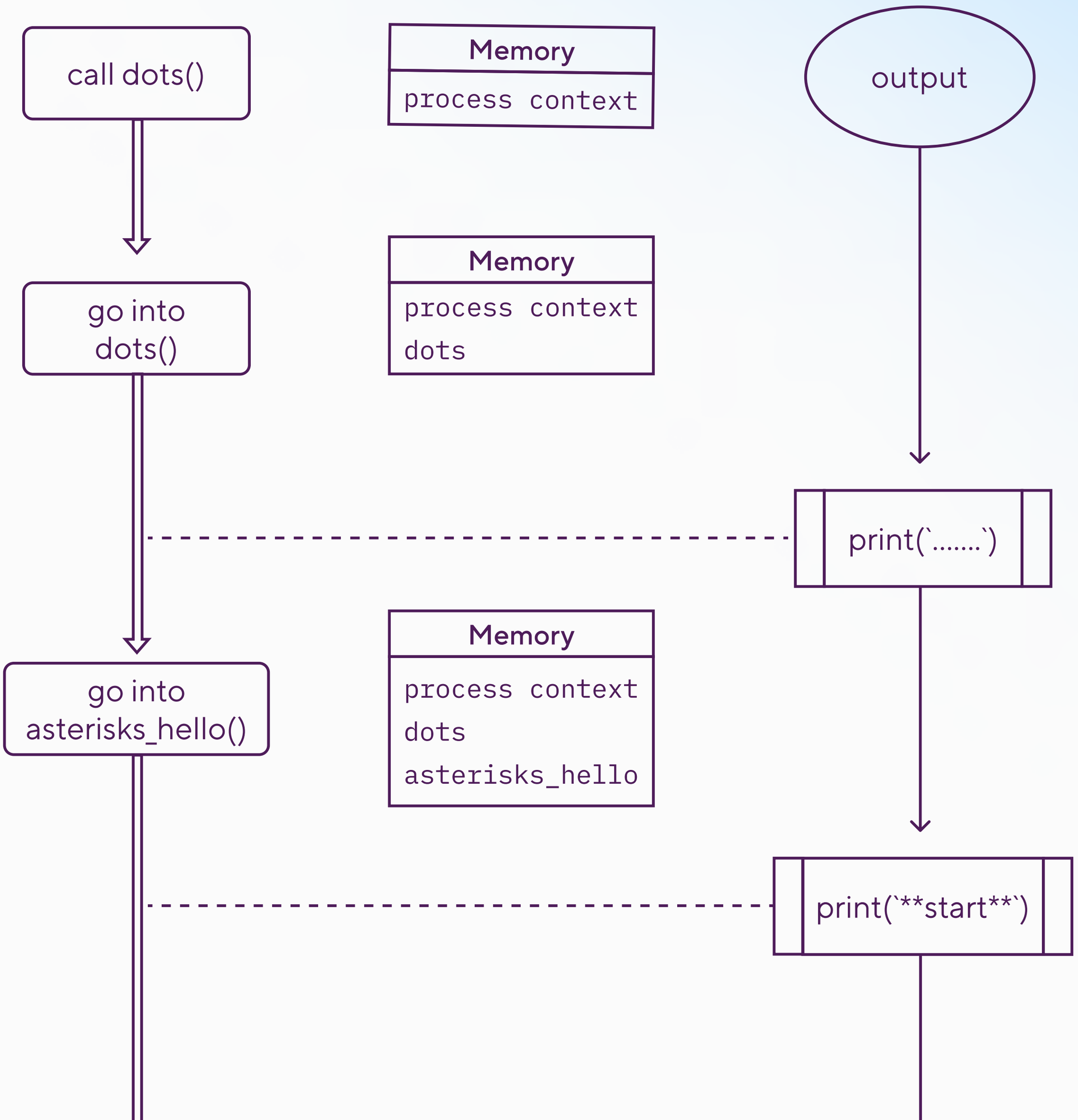
ooo | код

```
def hello():
    print('hello')

def asterisks_hello():
    print('**start**')
    hello()
    print('***end***')


def dots():
    print('.........')
    asterisks_hello()
    print('.........')

dots()
```

ooo | вывод

```
>>> .........
>>> **start**
>>> hello
>>> ***end***
>>> .........
```

Flow процесса



go into
hello()

Memory

```
process context
dots
asterisks_hello
hello
```

print(`hello`)

return to
asterisks_hello()

Memory

```
process context
dots
asterisks_hello
```

print(`**end**`)

return to
asterisks_hello()

Memory
`process context`
`dots`

print(`.....`)

return to
process context

Memory
`process context`

РЕКУРСИЯ

это процесс, в котором функция вызывает себя сама

Функции

ooo | код

```
def factorial(n):
    if not isinstance(n, int) or n < 0:
        raise AttributeError
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(5))
```

ooo | вывод

```
>>> 120
>>> 1 * 2 * 3 * 4 * 5 = 120
```

Функции

ooo | код

```
def factorial(n):
    if not isinstance(n, int) or n < 0:
        raise AttributeError
    if n == 0:
        return 1
    return n * factorial(n - 1)

def factorial_typing(n: int) -> int:
    if not isinstance(n, int) or n < 0:
        raise AttributeError
    if n == 0:
        return 1
    return n * factorial_typing(n - 1)

print(factorial(5))
```

ooo | вывод

```
>>> 120
```

Функции

ooo | код

```
def fibonacci(n):
    if not isinstance(n, int) or n < 0:
        raise AttributeError
    if n in (0, 1):
        return n
    return fibonacci(n - 2) + fibonacci(n - 1)

print(fibonacci(7))
```

ooo | вывод

```
>>> 13
>>> 0 1 1 2 3 5 8 13
```

ПЕРЕПОЛНЕНИЕ СТЕКА

происходит при заполнении стеком вызовов всей оперативной памяти

Плюсы Рекурсии

1. Простая логика для вычисления объектов
2. Улучшение читаемости кода

Минусы Рекурсии

1. Переполнение оперативной памяти
2. Неэффективность решения по времени

ЭЛИМИНАЦИЯ

сведение хвостовой рекурсии к циклу

Функции

ooo | код

```
def factorial_cycle(n):
    if not isinstance(n, int) or n < 0:
        raise AttributeError
    if n == 0:
        return 1
    mult = 1
    while n > 1:
        mult *= n
        n -= 1
    return mult

print(factorial_cycle(5))
```

ooo | вывод

```
>>> 120
```