Начнем в 20:01 [мск]

Что будет на экране?

```
x = 1
y = x / 3
z = y * 3

if x == z:
   print("Trump")
else:
   print("Biden")
```

# ML Basic

## Коллекции

otus.ru

Меня не видно?
Не слышно?
Запись не ведется?


Пишите в чат
или кричите


**Проверить, идет ли запись**

# Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

# ML Basic

## Коллекции

Катя

**Екатерина Анатольевна Дмитриева**

**Telegram:** @dmi3eva

# Что будет на экране?

```
x = 1
y = x / 3
z = y * 3

if x == z:
    print("Trump")
else:
    print("Biden")
```

# Что будет на экране?

```
x = 1
y = x / 3 # 0.333333333
z = y * 3

if x == z:
    print("Trump")
else:
    print("Biden")
```

# Что будет на экране?

```
x = 1
y = x / 3 # 0.333333333
z = y * 3 # 0.999999999

if x == z:
    print("Trump")
else:
    print("Biden")
```

# Посложнее

```
from math import sin, asin, sqrt

x = 1
y = sin(sin(x) ** 2)
z = asin(sqrt(asin(y)))  # 1.0000000000000002

if x == z:
 print("Trump")
else:
 print("Biden")
```

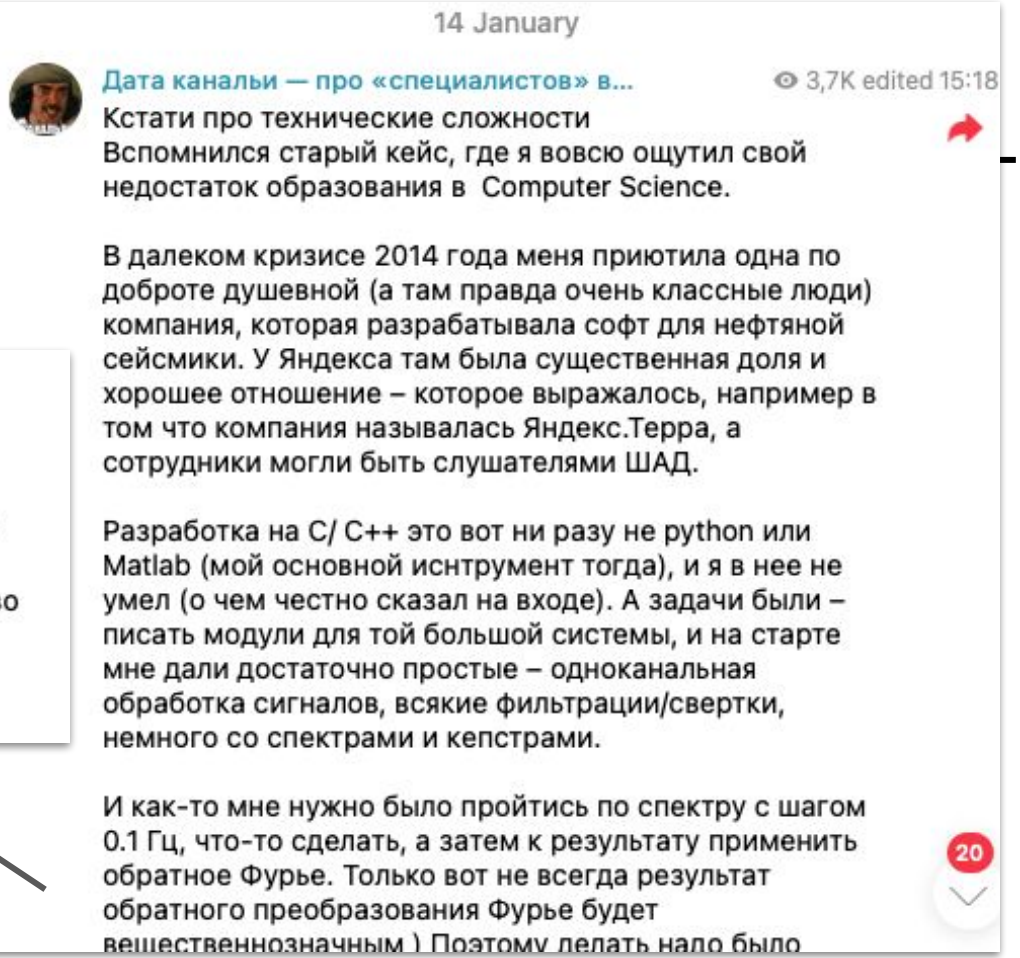
# Что будет на экране?

```
x = 1
y = x / 3 # 0.333333333
z = y * 3 # 0.999999999

if x == z:
    print("Trump")

EPS = 0.000000001

if abs(x - z) < EPS:
    print("Trump")
```

# История

https://t.me/datarascals/116

Несколько дней я потратил, пытаясь добиться того же результата в C – без шанса 🙈🤯.
В матлабе же не только индексация массивов отличается)
В итоге пошел на поклон к синьору и тут вскрылся мой недостаток образования на тот момент в CS. Что-то о свойствах вещественных чисел я знал (что на равенство сравнивать нельзя, ибо хранятся они в некотором приближении), но вот глубоко не копал – на чем и погорел.



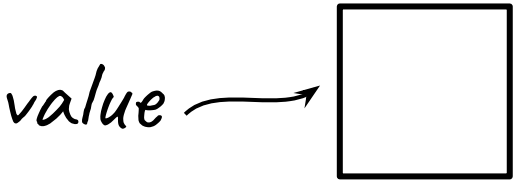
# ПОВТОРЕНИЕ ЦИКЛОВ

# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```

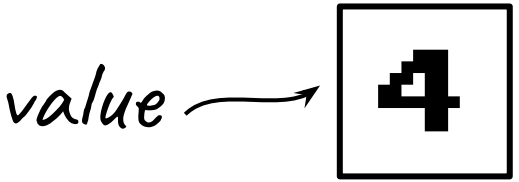
# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



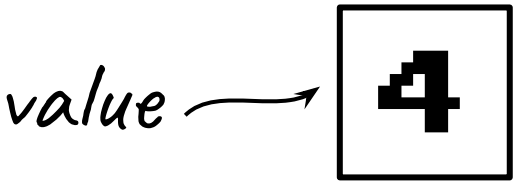
# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



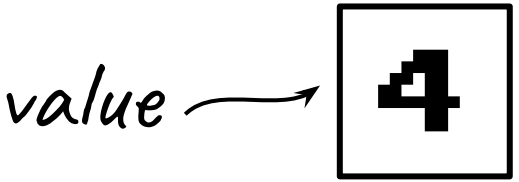
# Разбор №1

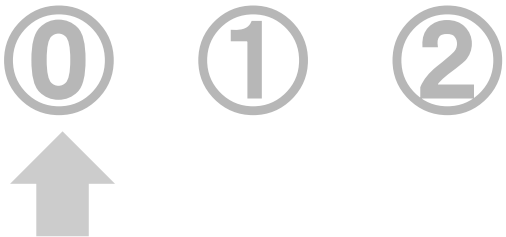
```
value = 4
for i in range(3):
    value = value + 1
print(value)
```




# Разбор №1

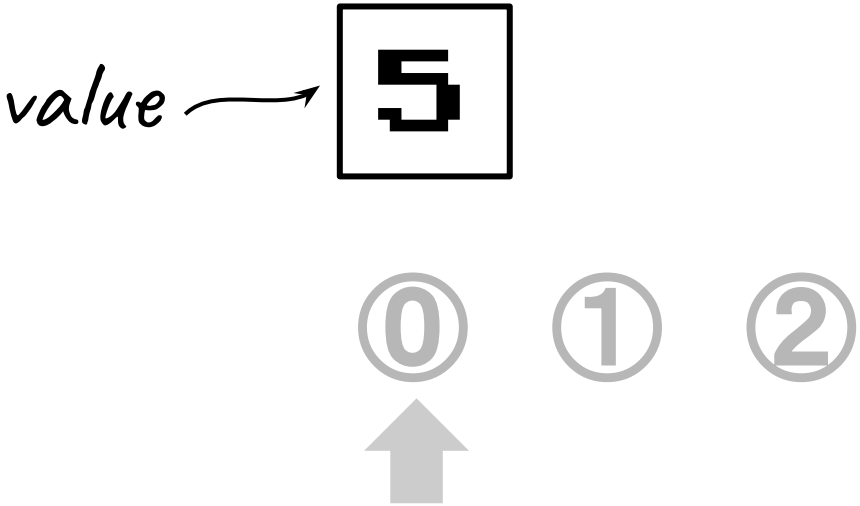
```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



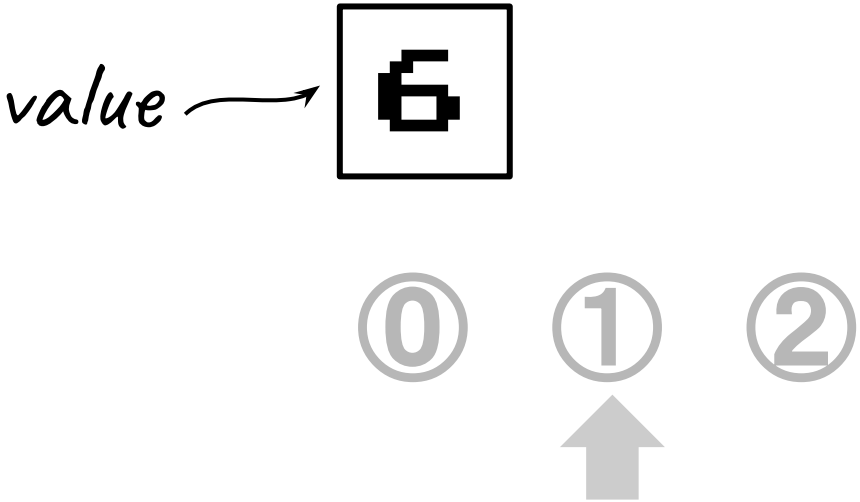
# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



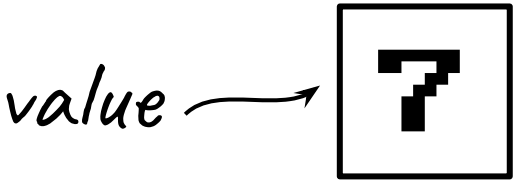
# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



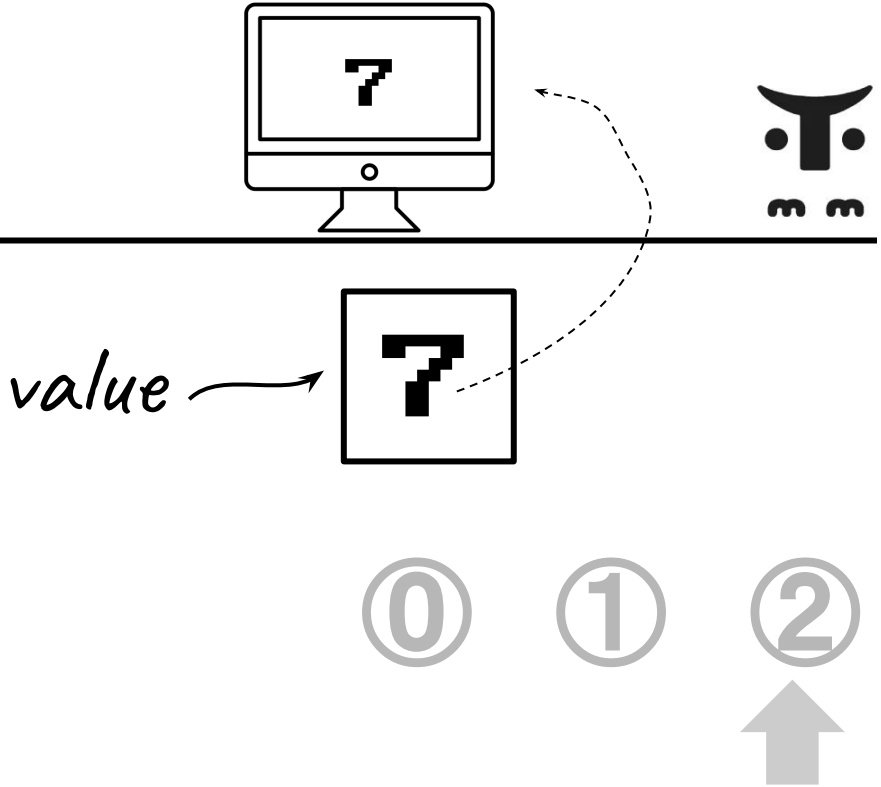
# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



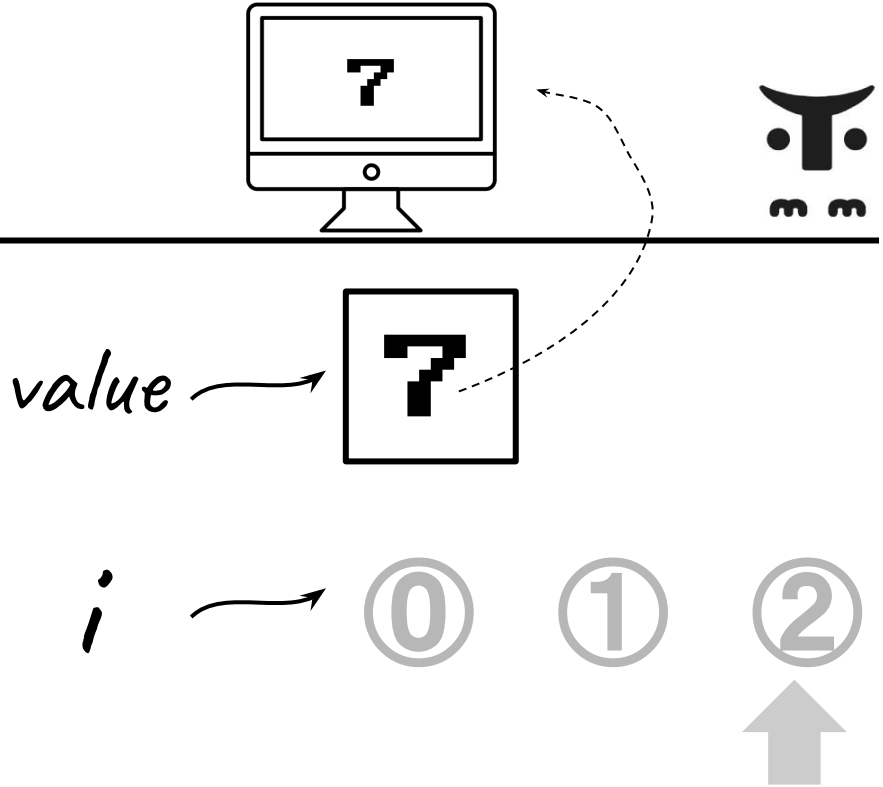

# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



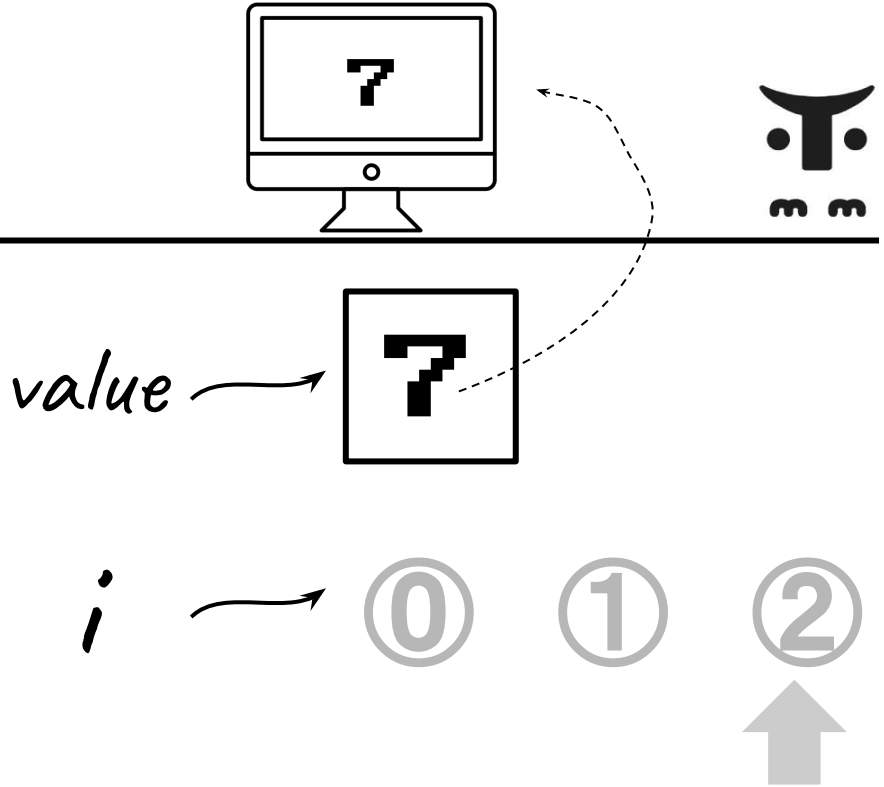
# Разбор №1

```
value = 4
for i in range(3):
    value = value + 1
print(value)
```



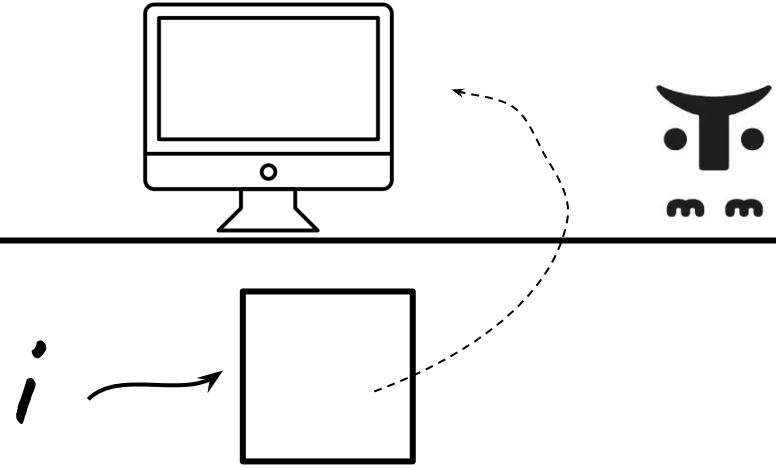
# Разбор №1

```
value = 4   i = 0
for i in range(3):
    value = value + 1
    i = i + 1
print(value)
```



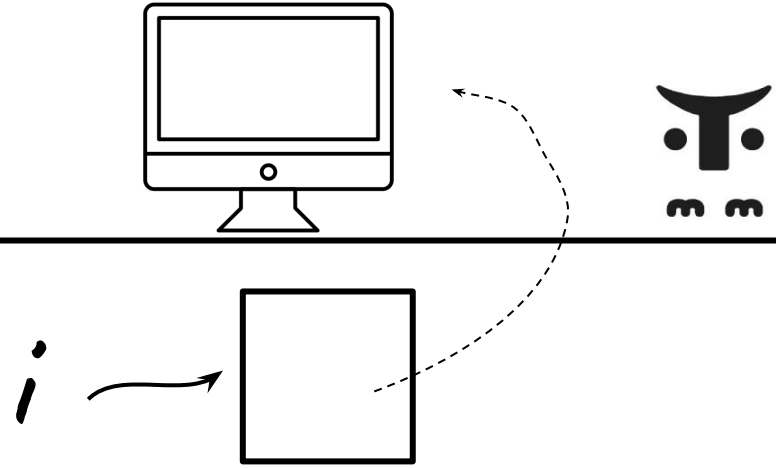
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



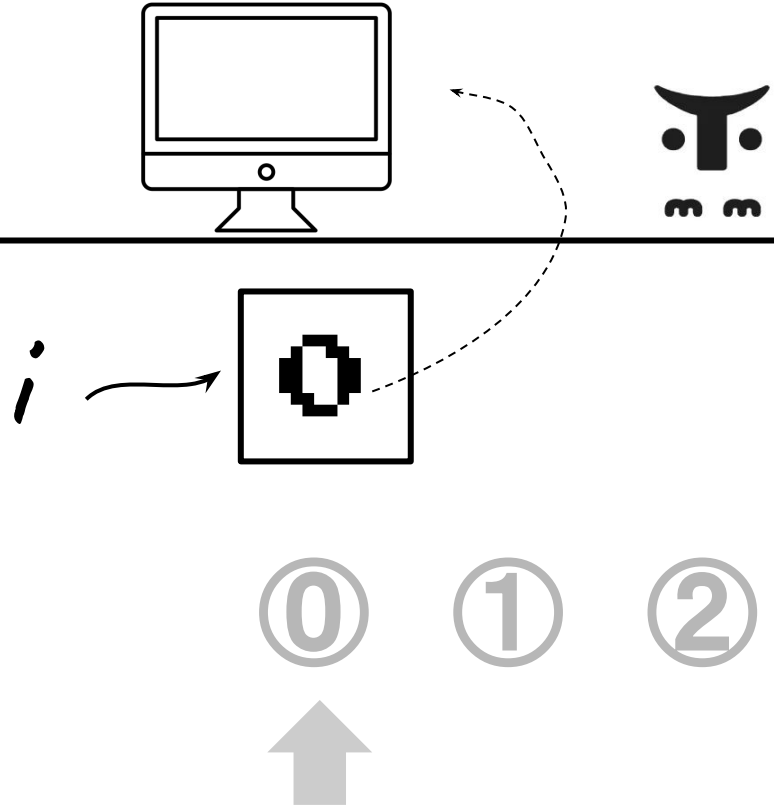
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



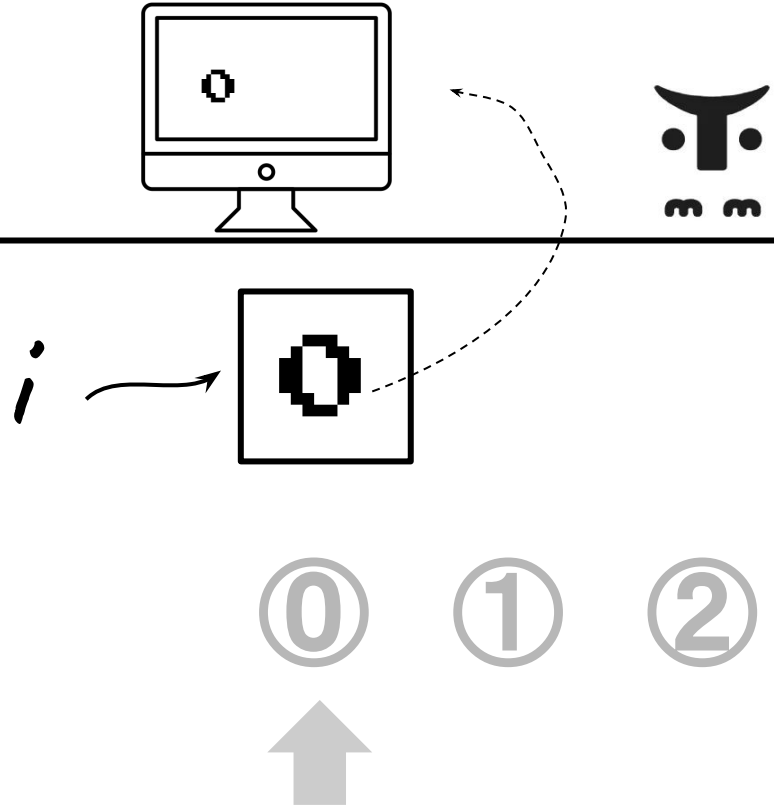
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



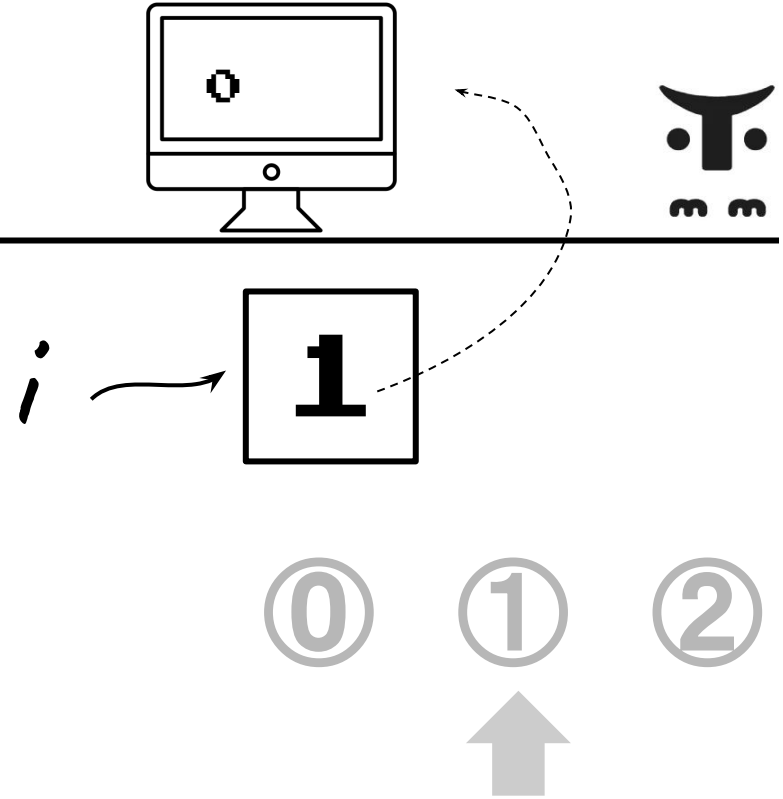
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



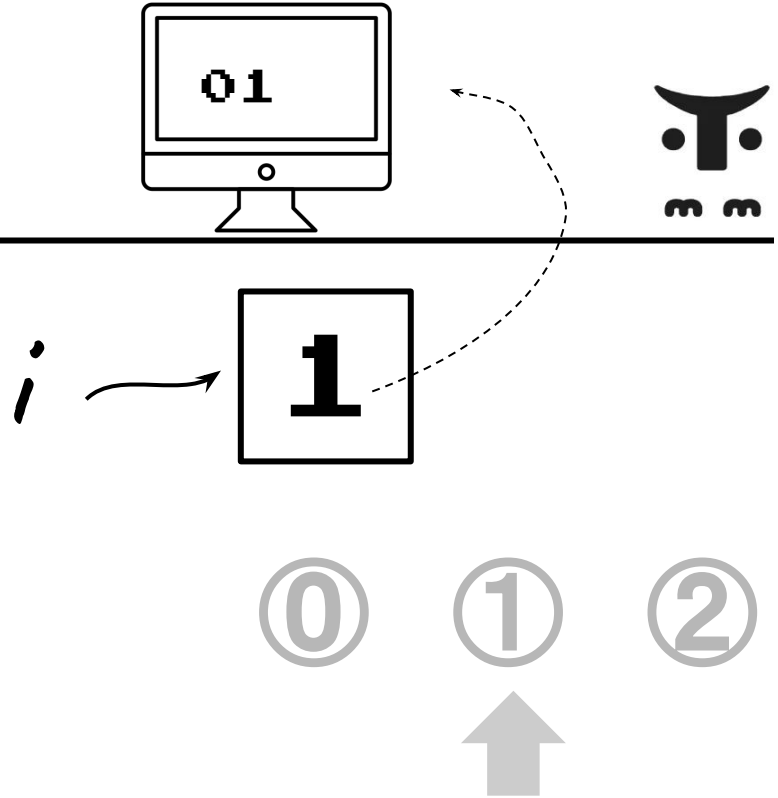
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



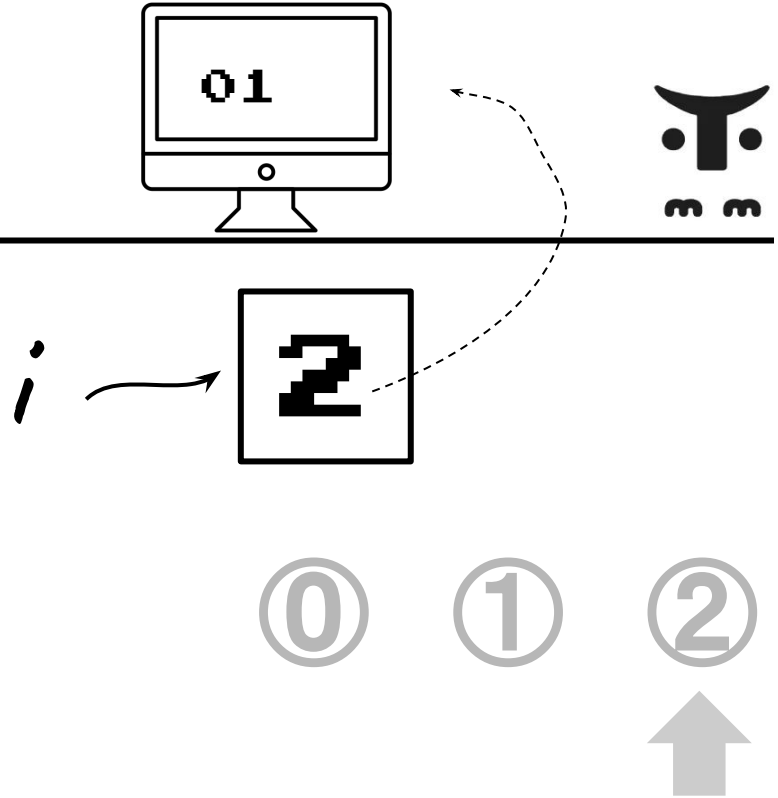
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



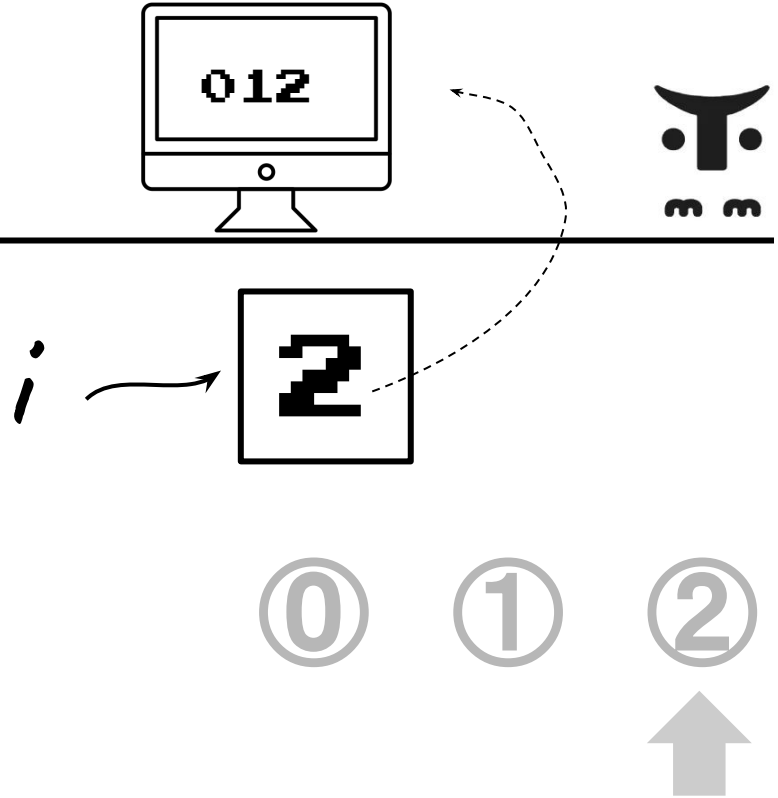
# Разбор №2

```
for i in range(3):
    print(i, end="")
```



# Разбор №2

```
for i in range(3):
    print(i, end="")
```

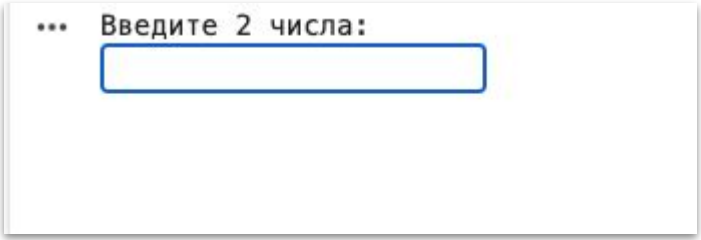


# РАБОТА СО СПИСКАМИ

# Учебные задания

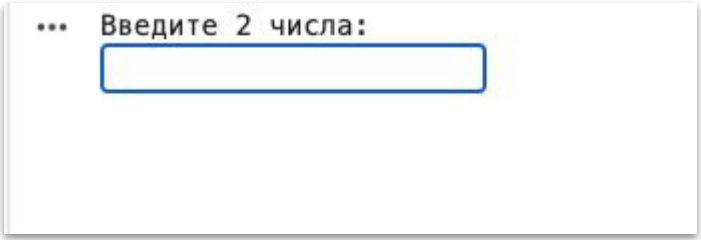
# Реальная жизнь

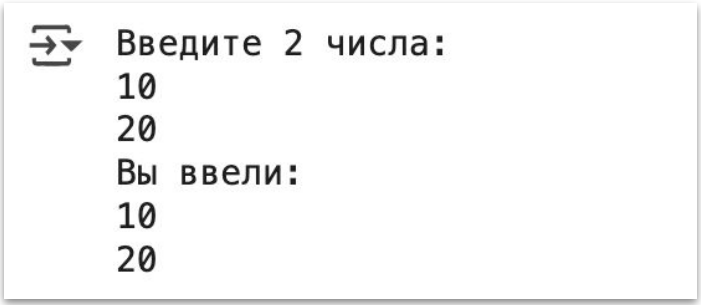
Считайте 2 числа и выведите их



# Учебные задания Реальная жизнь

Считайте 2 числа и выведите их



# Учебные задания

# Реальная жизнь

Считайте 2 числа и выведите их

```
a = int(input())
b = int(input())
```

# Учебные задания Реальная жизнь

Считайте 2 числа и выведите их

```
a = int(input())
b = int(input())
print(a)
print(b)
```

# Учебные задания Реальная жизнь

Считайте **3** числа и выведите их

```
a = int(input())
b = int(input())
print(a)
print(b)
```

# Учебные задания Реальная жизнь

Считайте **3** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
print(a)
print(b)
print(c)
```

# Учебные задания Реальная жизнь

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(a)
print(b)
print(c)
print(d)
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(a)
print(b)
print(c)
print(d)
```

# Реальная жизнь

Считайте **миллион** чисел, …

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
…
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(a)
print(b)
print(c)
print(d)
```

# Реальная жизнь

Считайте **n** чисел, …

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
…
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(a)
print(b)
print(c)
print(d)
```

# Реальная жизнь

Считайте **100** чисел, …

- **Циклы**, чтобы производить много одинаковых действий автоматически

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
```

# Реальная жизнь

Считайте **100** чисел, …

- **Циклы**, чтобы производить много одинаковых действий автоматически

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
```

# Реальная жизнь

Считайте **100** чисел, …

- **Циклы**, чтобы производить много одинаковых действий автоматически

```
for i in range(100):
    ??? = int(input())
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(a)
print(b)
print(c)
print(d)
```

# Реальная жизнь

Считайте **100** чисел, …

- **Циклы**, чтобы производить много одинаковых действий автоматически

```
for i in range(100):
    ??? = int(input())
```

```
for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(a)
print(b)
print(c)
print(d)
```

# Реальная жизнь

Считайте **100** чисел, …

- **Списки**, чтобы хранить сразу много значений

```
for i in range(100):
    ??? = int(input())

for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

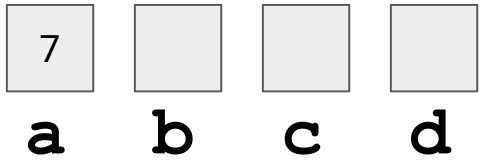
- **Списки**, чтобы хранить сразу много значений

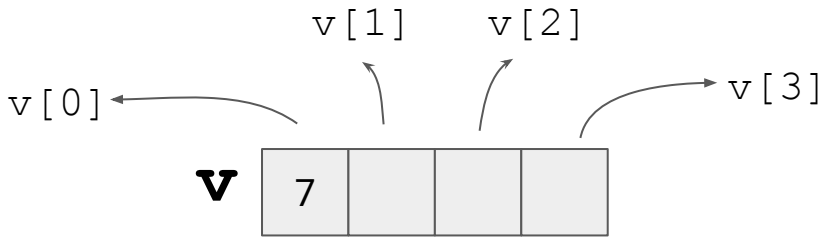
```
for i in range(100):
    ??? = int(input())

for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

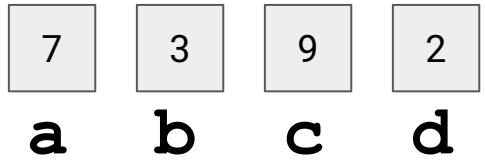


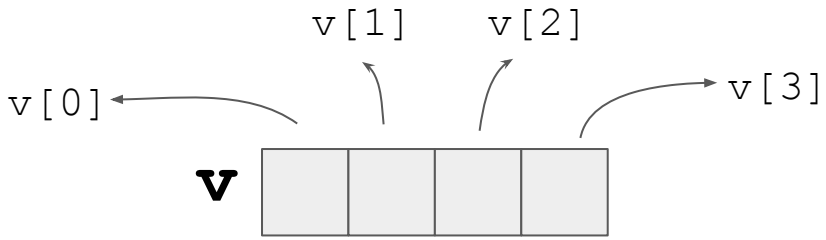



# Учебные задания

Считайте **4** числа и выведите их

```
a = 7
b = 3
c = 9
d = 2
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

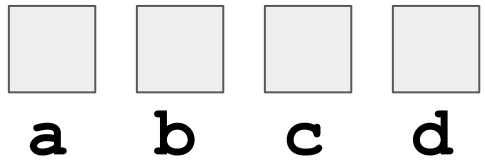


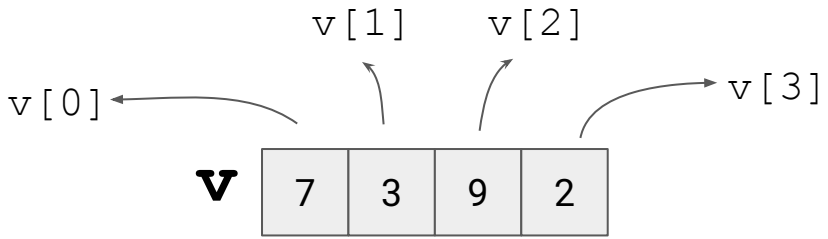



# Учебные задания

Считайте **4** числа и выведите их

```
v = [7, 3, 9, 2]
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

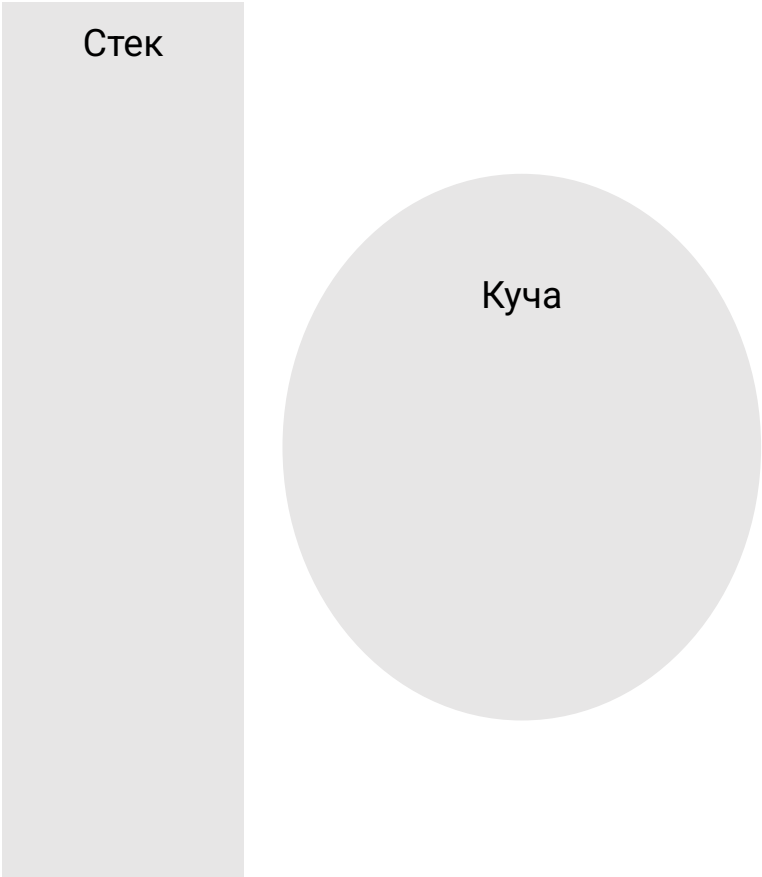


# Учебные задания

Считайте **4** числа и выведите их

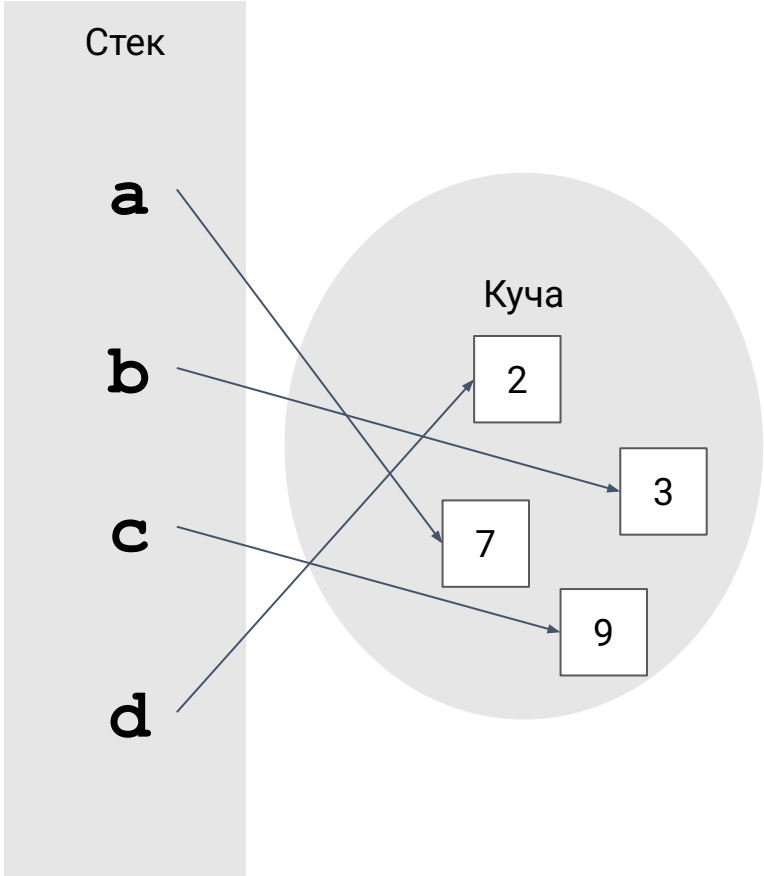
```
v = [7, 3, 9, 2]
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Учебные задания

Считайте **4** числа и выведите их

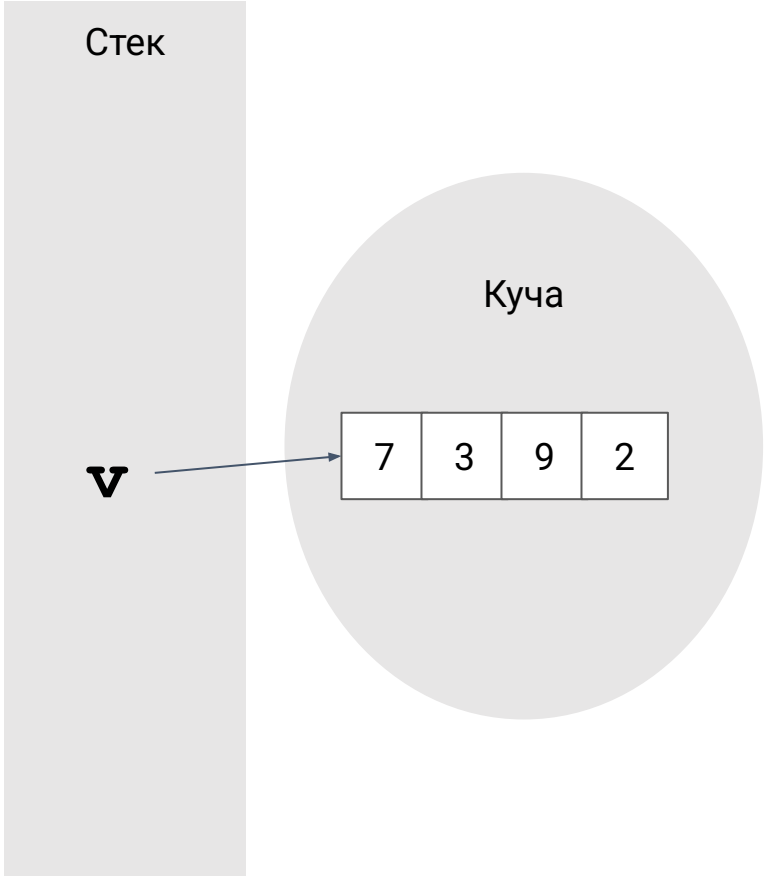
```
a = 7
b = 3
c = 9
d = 2
print(a)
print(b)
print(c)
print(d)
```



# Учебные задания

Считайте **4** числа и выведите их

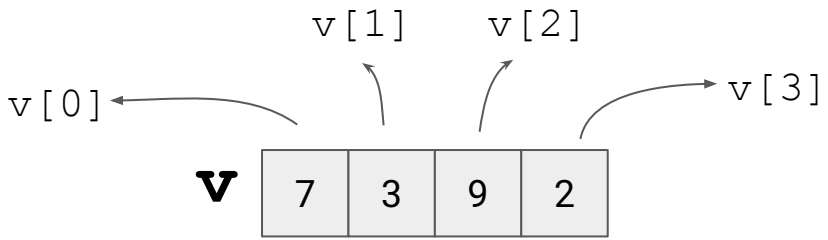
```
v = [7, 3, 9, 2]
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

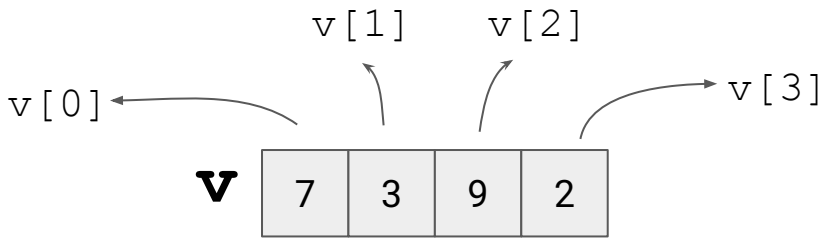
```
v = [7, 3, 9, 2]
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

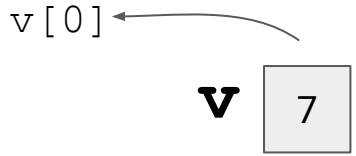
```
v = []
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

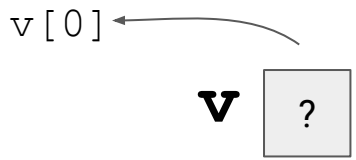
```
v = []
v.append(7)
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

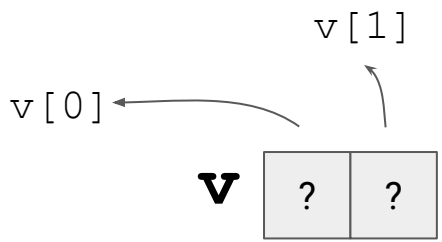
```
v = []
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

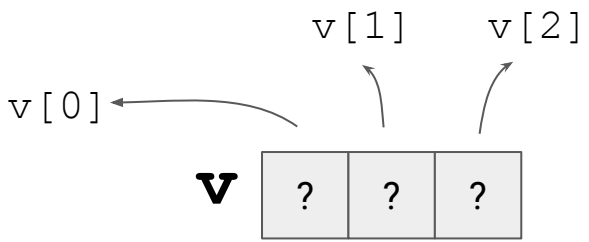
```
v = []
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

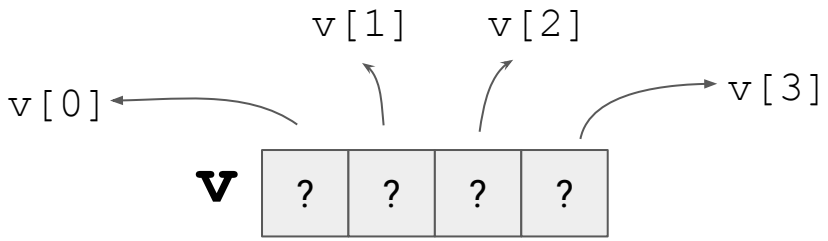
```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Но мы хотим считать!

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```



# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
for i in range(100):
    ???
```

```
for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    ???

for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    ???

for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    v.append(int(input()))

for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    v.append(int(input()))

for i in range(100):
    print(???)
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    v.append(int(input()))

for i in range(100):
    print(v[???])
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    v.append(int(input()))

for i in range(100):
    print(v[x])
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    v.append(int(input()))
x = 0
for i in range(100):
    print(v[x])
    x = x + 1
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
  v.append(int(input()))

for i in range(100):
  print(v[i])
```

# Учебные задания

Считайте **4** числа и выведите их

```
v = []
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

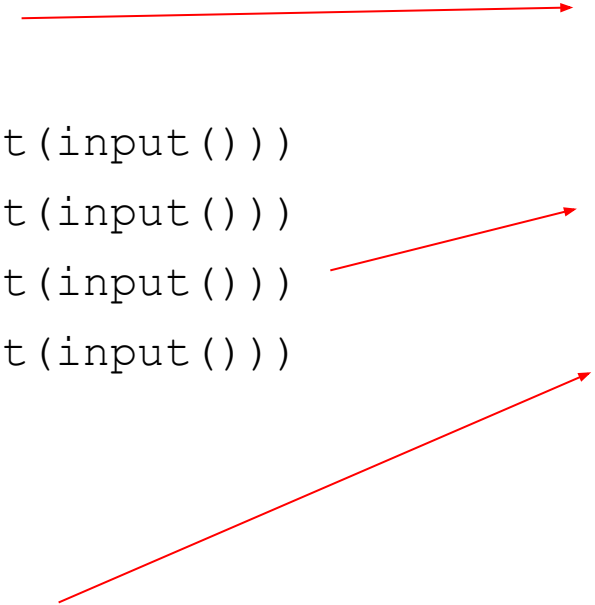
# Реальная жизнь

Считайте **100** чисел, …

```
v = []

for i in range(100):
    tmp = int(input())
    v.append(tmp)

for i in range(100):
    print(v[i])
```

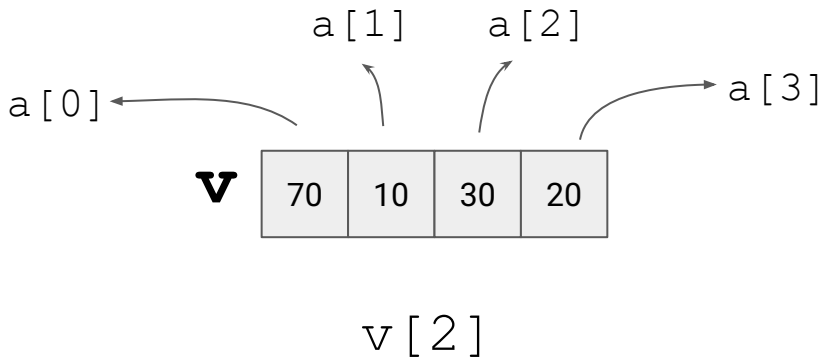
# С циклами и без

```
v = []


v.append(int(input()))
v.append(int(input()))
v.append(int(input()))
v.append(int(input()))


print(v[0])
print(v[1])
print(v[2])
print(v[3])
```

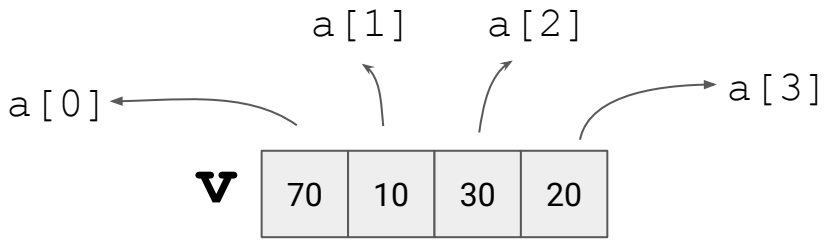


```
v = []

for i in range(4):
  tmp = int(input())
  v.append(tmp)

for i in range(4):
  print(v[i])
```

# Индекс vs. значение



# Индекс vs. значение




v[2]

**2** – индекс

**30** – значение

# статус

| | List | Tuple | Dict | Set |
|---|---|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | | | |
| Инициализация | `elements = [200, 300]`<br>`elements = "2+3".split("2+3")` | | | |
| Обращение к элементу | `v = elements[1]`<br>`v = elements[1:3]` | | | |
| Добавление | `elements.append(400)` | | | |
| Расширение | `elements += [500, 600]` | | | |
| Изменение | `elements[7] = "bob"`<br>`elements[1:2] = [0, 0]` | | | |
| Проверка наличия | `if 200 in elements:` | | | |
| Comprehension | `e = [_a for _a in elements]` | | | |

| | List | Tuple | Dict | Set |
|---|---|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | | | |
| Инициализация | `elements = [200, 300]`<br>`elements = "2+3".split("2+3")` | | | |
| Обращение к элементу | `v = elements[1]`<br>`v = elements[1:3]` | | | |
| Добавление | `elements.append(400)` | | | |
| Расширение | `elements += [500, 600]` | | | |
| Изменение | `elements[7] = "bob"`<br>`elements[1:2] = [0, 0]` | | | |
| Проверка наличия | `if 200 in elements:` | | | |
| Comprehension | `e = [_a for _a in elements]` | | | |

SPLIT

7. Способы задания списка:

```
values = [10, 30, 20]
values = [10, "ten", 2.4]
values = []
values.append(10)
values = "one+two+three".split("+") # -> ['one', 'two', 'three']
```

# split()

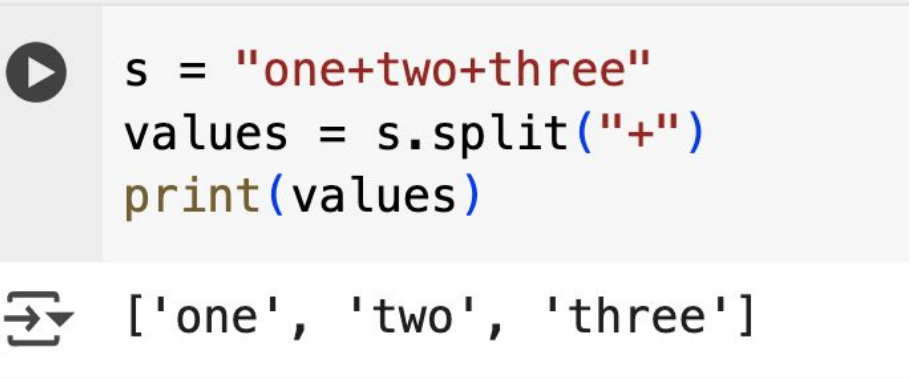
Строка → Список строк

```
s = "one+two+three"

values = s.split("+")

print(values)
```

# split()

Строка → Список строк

```
s = "one+two+three"

values = s.split("+")

print(values)
```



# Сколько будет элементов?

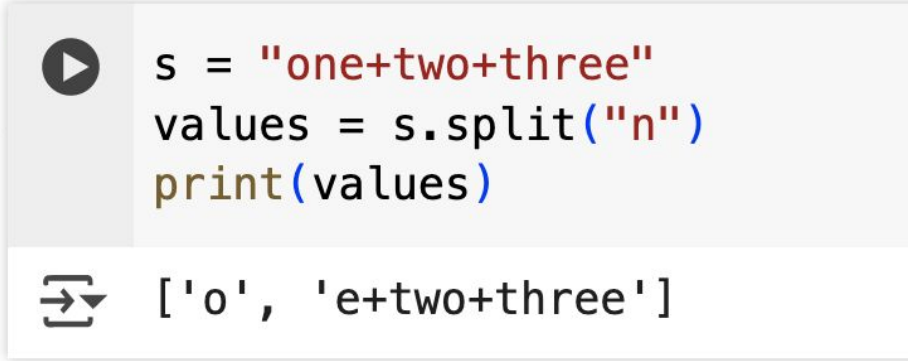
Строка → Список строк

```
s = "one+two+three"

values = s.split("n")

print(values)
```

# Сколько будет элементов?

Строка → Список строк

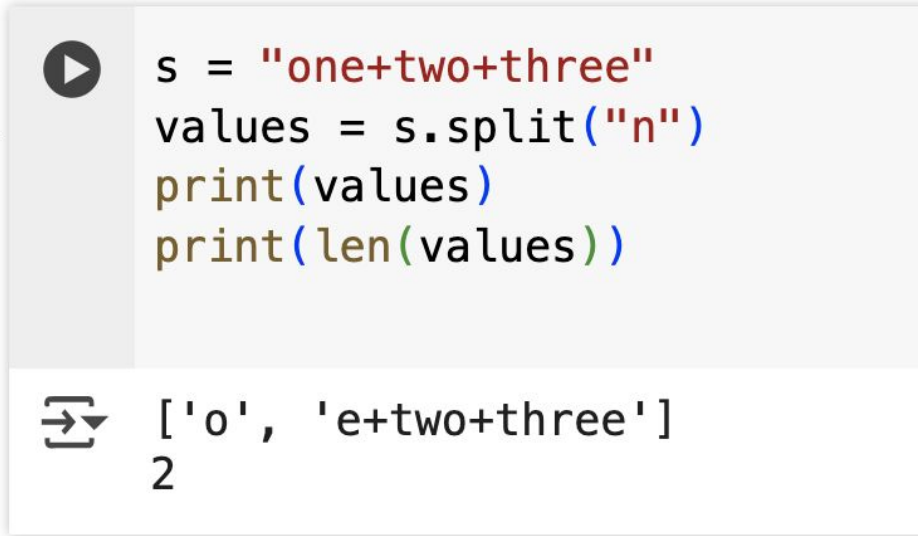
```
s = "one+two+three"

values = s.split("n")

print(values)
```



# Сколько будет элементов?

Строка → Список строк

```
s = "one+two+three"

values = s.split("n")

print(values)

print(len(values))
```



# Сколько будет элементов?

Строка → Список строк

```
s = "one+two+three"

values = s.split("n")

print(values)

l = len(values)

print(l)
```

```
s = "one+two+three"
values = s.split("n")
print(values)
print(len(values))
```

```
['o', 'e+two+three']
2
```

# СРЕЗЫ

# Срез

```
menu = ["bacon", "egg", "coffee", "croissant", "doshirak"]

breakfast = menu[2:4]
```

# Срез

```
menu = ["bacon", "egg", "coffee", "croissant", "doshirak"]

breakfast = menu[2:4]
```

# Срез

```
menu = ["bacon", "egg", "coffee", "croissant", "doshirak"]

breakfast = menu[2:]
```

# Срез

```
menu = ["bacon", "egg", "coffee", "croissant", "doshirak"]

breakfast = menu[:4]
```

# Срез

```
menu = ["bacon", "egg", "coffee", "croissant", "doshirak"]

breakfast = menu[:]
```

# ПРАКТИКА

# Что делать?

- https://docs.google.com/presentation/d/1Y6zULhpOT7Skd8yN26C4_kVsmNhVb6rMbHx-HlQrUo4/edit?usp=sharing
- Памятка: https://clck.ru/3GEng7
- Вопросы: https://t.me/dmi3eva

**[Коротко: https://goo.su/yqSdwF]**

# Как перебрать элементы списка?

# Так мы перебирали элементы массива:

```
values: [20, 19, 10]
```

**Pascal:**

```
for i := 1 to 3 do
begin
    writeln(values[i]);
end;
```

**C:**

```
for (i = 0; i < 3; i++){
    printf("%d\n", values[i]);
}
```

# Также можно и в Python:

```
values: [20, 19, 10]
```

**Python:**

```
for i in range(???):
    print(values[i])
```

# Также можно и в Python:

```
values: [20, 19, 10]
```

**Python:**

```
for i in range(0, 3):
    print(values[i])
```

# Также можно и в Python:

```
values: [20, 19, 10]
```

**Python:**

```
for i in range(3):
    print(values[i])
```

# Также можно и в Python:

```
values: [20, 19, 10]
```

**Python:**

```
for i in range(len(values)):
    print(values[i])
```

# Но можно и нужно проще:

```
values: [20, 19, 10]
```

**Python:**

```
for i in range(len(values)):
    print(values[i])
```

# Но можно и нужно проще:

```
values: [20, 19, 10]
```

**Python:**

```
for _v in values:
    print(_v)
```

**Python:**

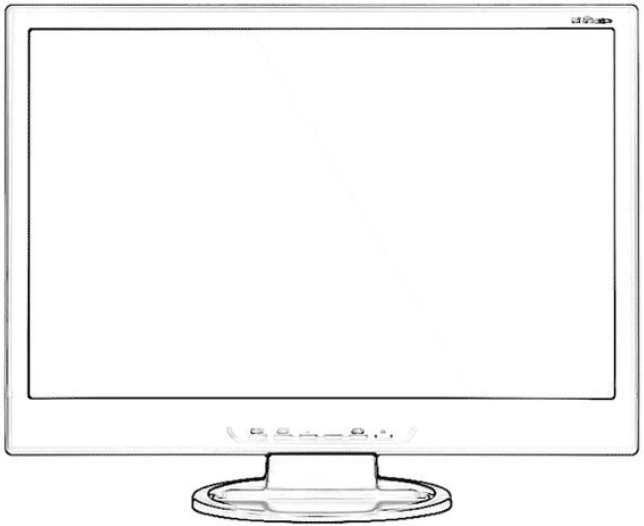
```
for i in range(len(values)):
    print(values[i])
```

# Но можно и нужно проще:

```
values = [20, 19, 10]
```

**Python:**

```
for _v in values:
    print(_v)
```

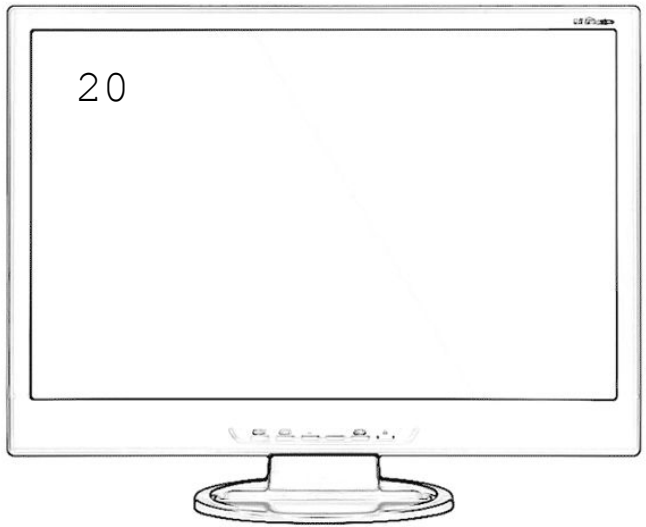


# Но можно и нужно проще:

**Python:**

```
for _v in values:
    print(_v)
```

```
values = [20, 19, 10]
           _v
```

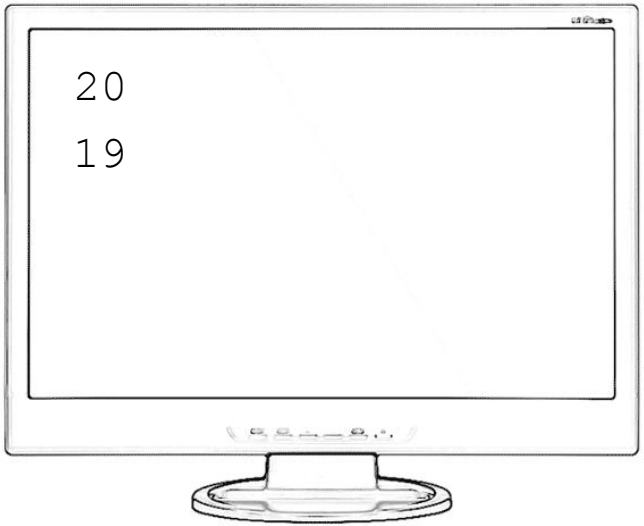


# Но можно и нужно проще:

**Python:**

```
for _v in values:
    print(_v)
```

```
values = [20, 19, 10]
           _v
```

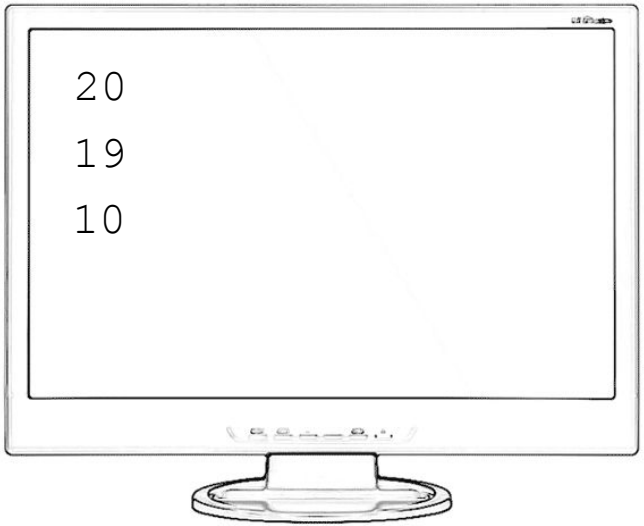


# Но можно и нужно проще:

**Python:**

```
for _v in values:
    print(_v)
```

```
values = [20, 19, 10]
           _v
```

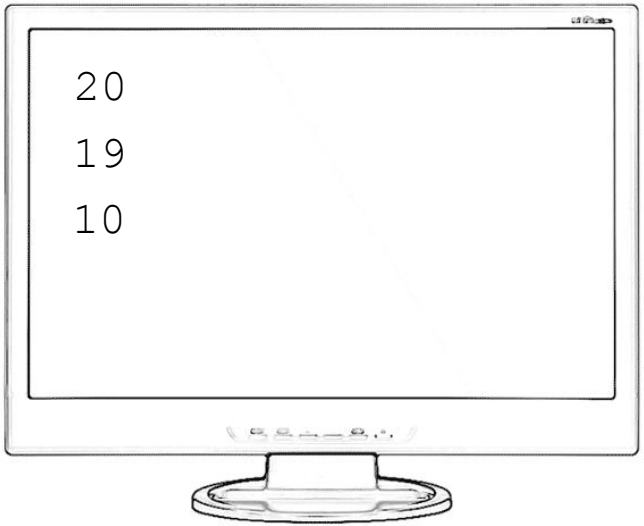


# Но можно и нужно проще:

**Python:**

```
for _v in values:
    print(_v % 10)
```

```
values = [20, 19, 10]
           _v
```

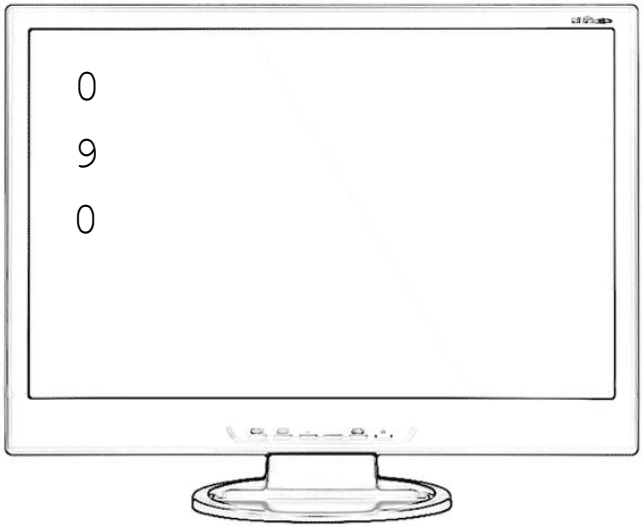


# Но можно и нужно проще:

**Python:**

```
for _v in values:
    print(_v % 10)
```

```
values = [20, 19, 10]
           _v
```

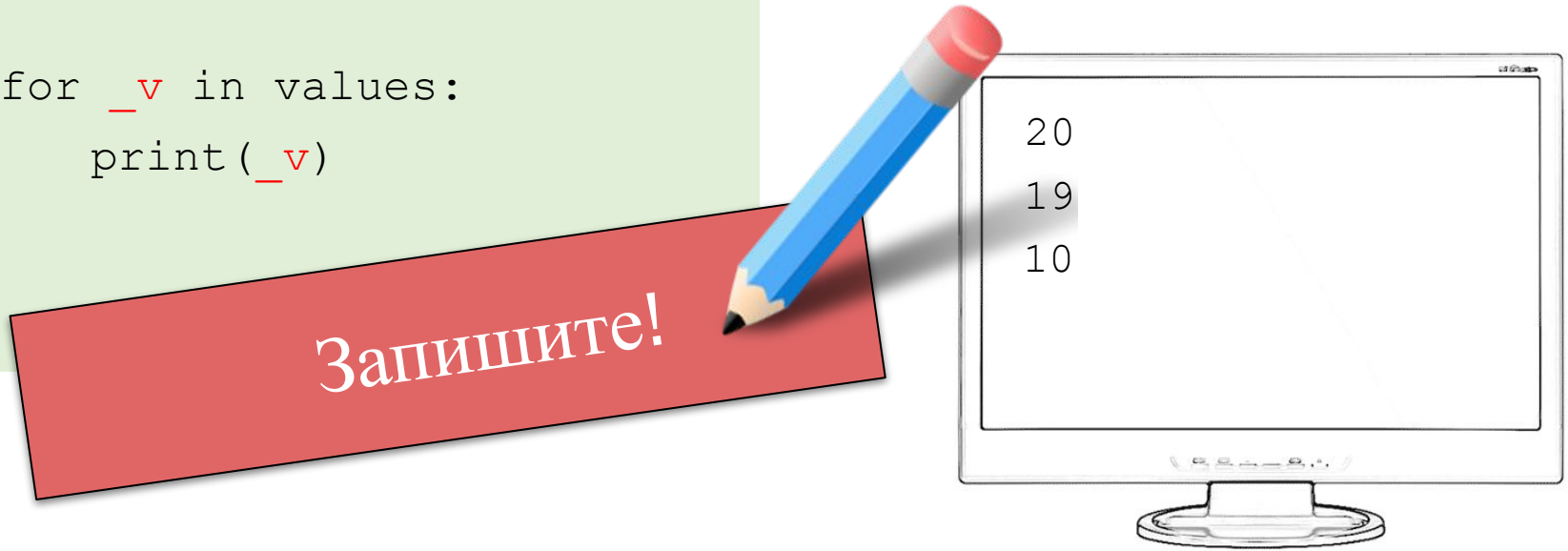


# Но можно и нужно проще:

**Python:**

```
for _v in values:
    print(_v)
```

```
values = [20, 19, 10]
           _v
```



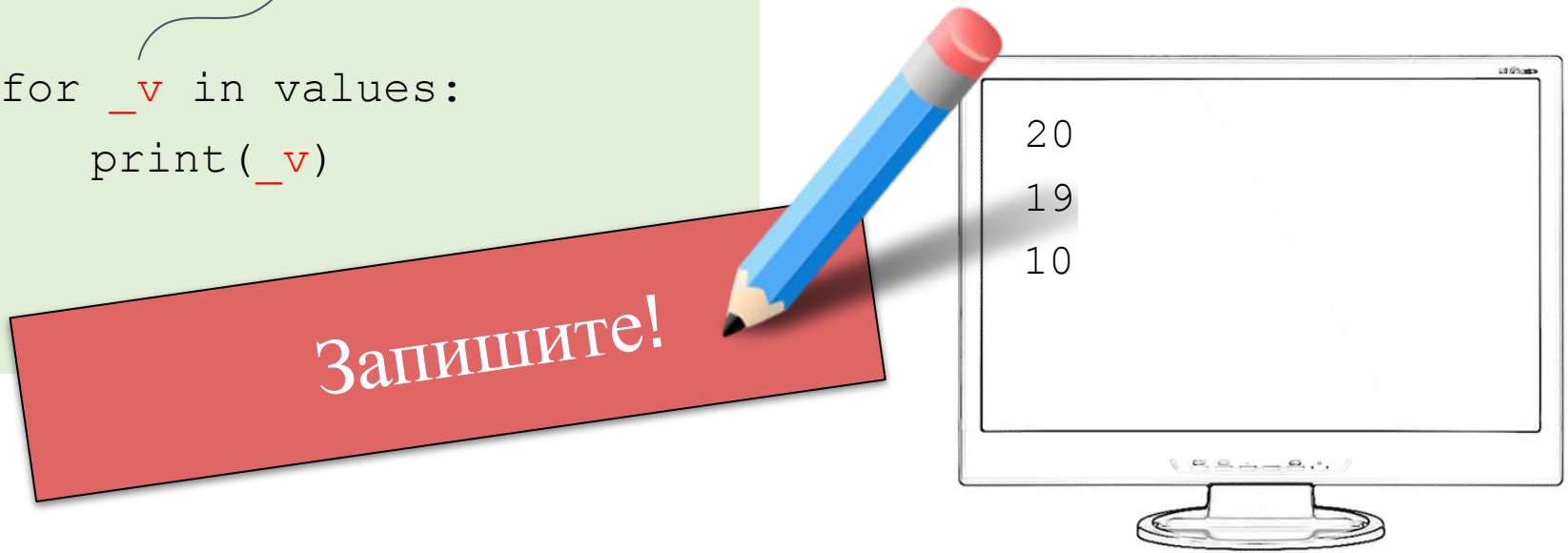
# Но можно и нужно проще:

Переменные цикла принято начинать с нижнего подчеркивания

**Python:**

```
for _v in values:
    print(_v)
```

```
values = [20, 19, 10]
           _v
```



# Что выведется на экран?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```

# Что выведется на экран?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```

# Что выведется на экран?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```

```
[3]
```

# Что выведется на экран?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```

```
[3]
```

# Что выведется на экран?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```

```
[3, 5]
```

# Что выведется на экран?

**Python:**

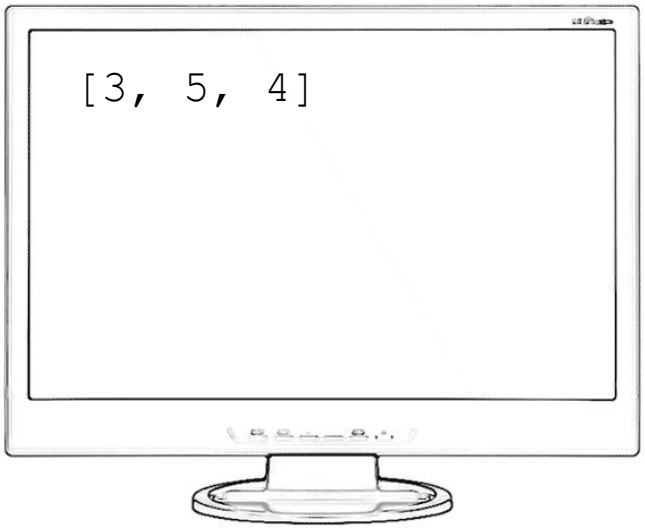
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```

```
[3, 5, 4]
```

# Что выведется на экран?

**Python:**

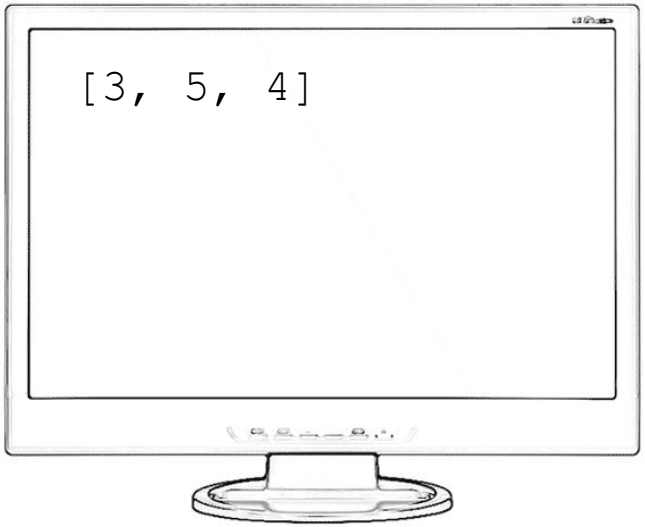
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```



# **animal** принимает значения элементов списка

**Python:**

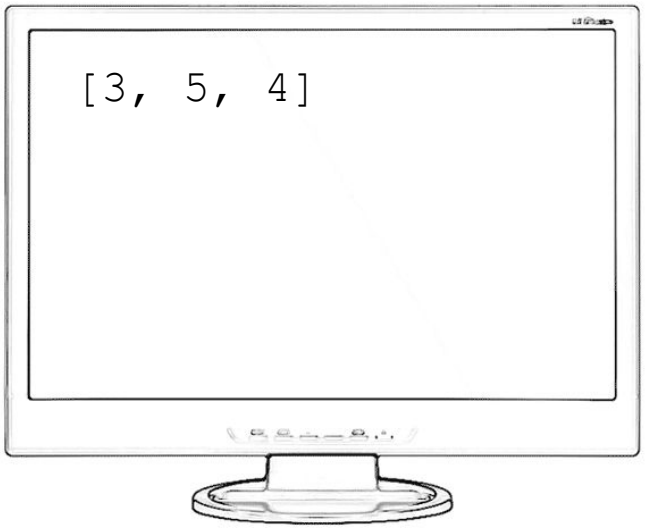
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _animal in zoo:
    numbers.append(len(_animal))
print(numbers)
```



# Можно называть по-другому

**Python:**

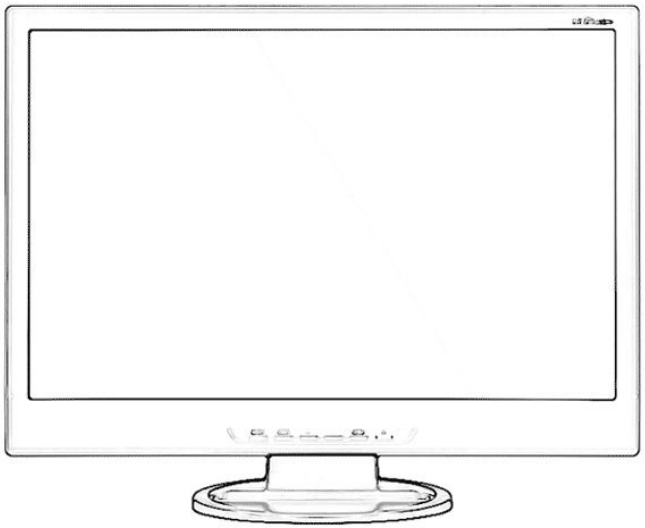
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(len(_x))
print(numbers)
```



# А, если так?

**Python:**

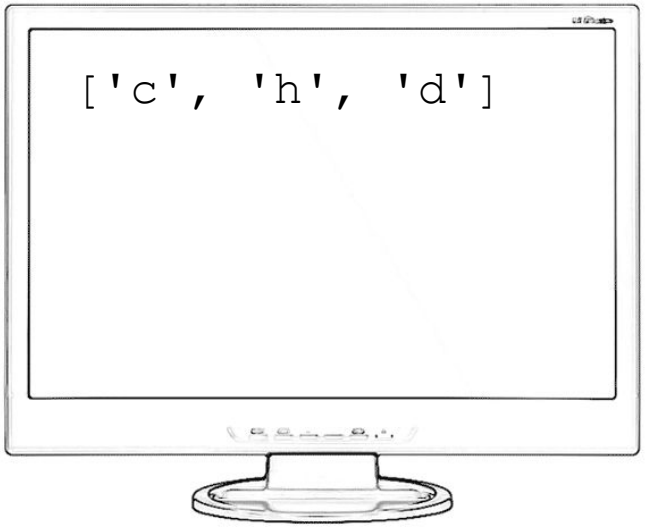
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(_x[0])
print(numbers)
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

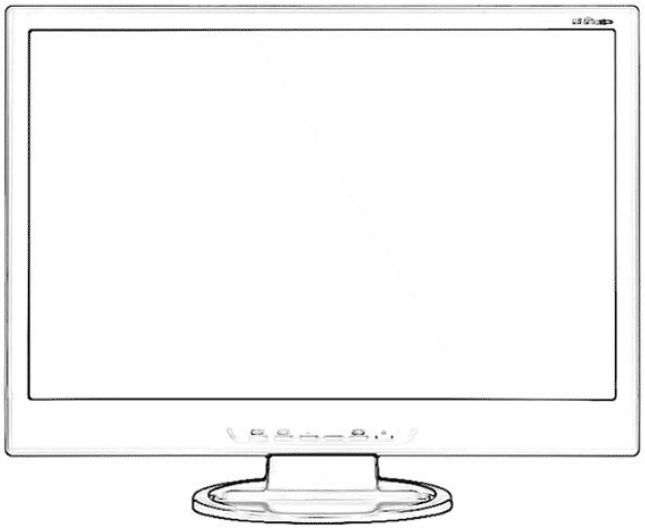
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(_x[0])
print(numbers)
```



# А, если так?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(len(zoo))
print(numbers)
```



# А, если так?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(len(zoo))
print(numbers)
```

```
[3]
```

# А, если так?

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(len(zoo))
print(numbers)
```

```
[3, 3]
```

# А, если так?

**Python:**

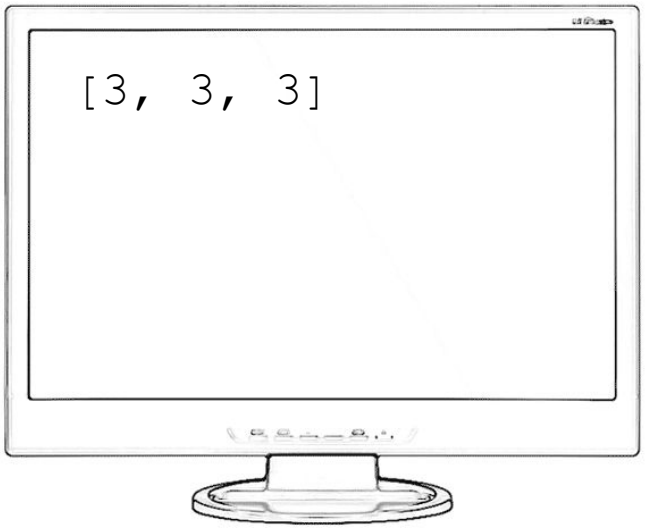
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(len(zoo))
print(numbers)
```

```
[3, 3, 3]
```

# А, если так?

**Python:**

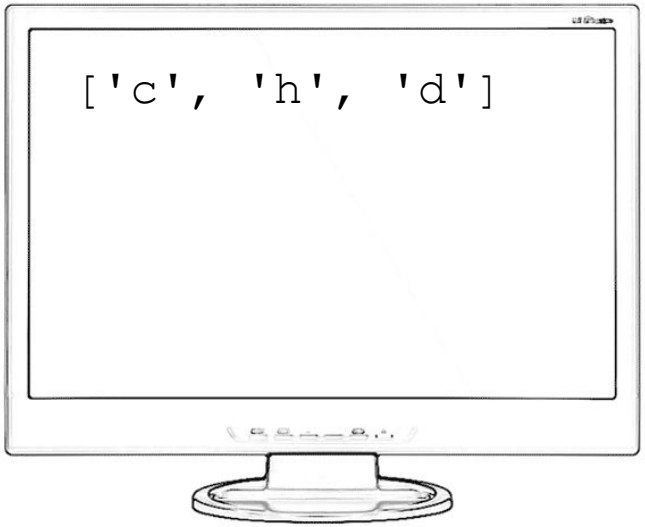
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(len(zoo))
print(numbers)
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

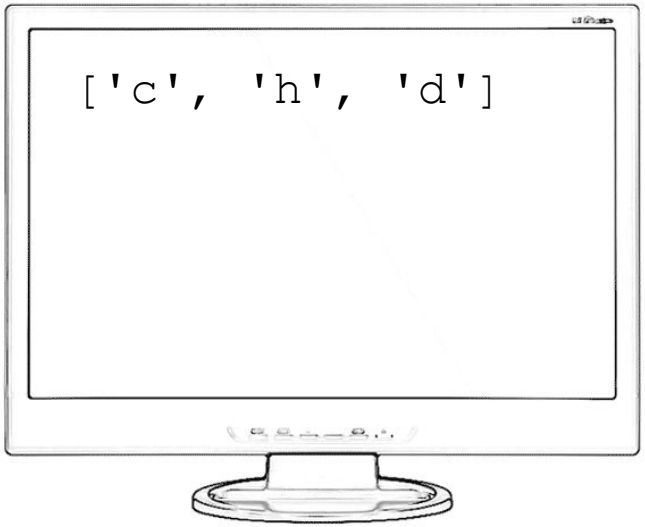
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(_x[0])
print(numbers)
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

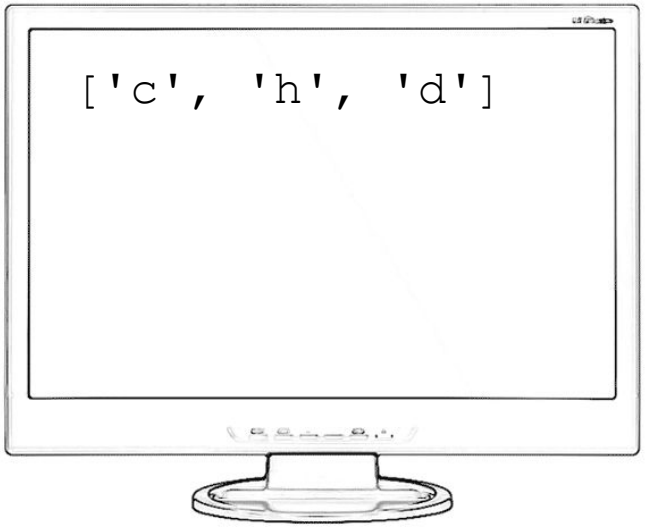
```
zoo = ['cat', 'hippo', 'bear']
numbers = []
for _x in zoo:
    numbers.append(_x[0])
print(numbers)
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

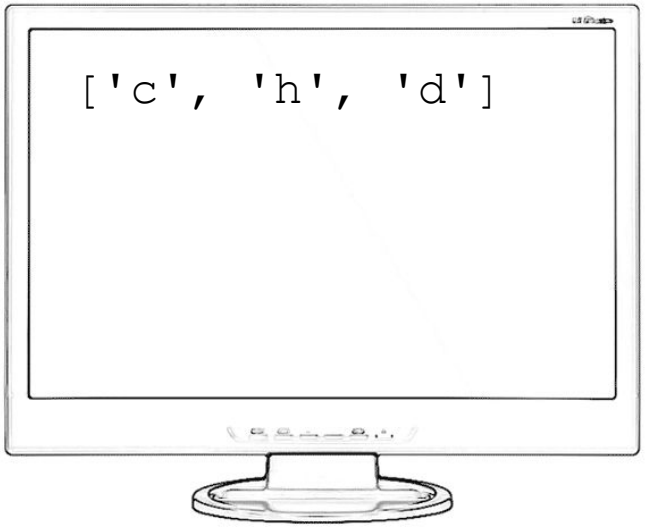
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
print(numbers)
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

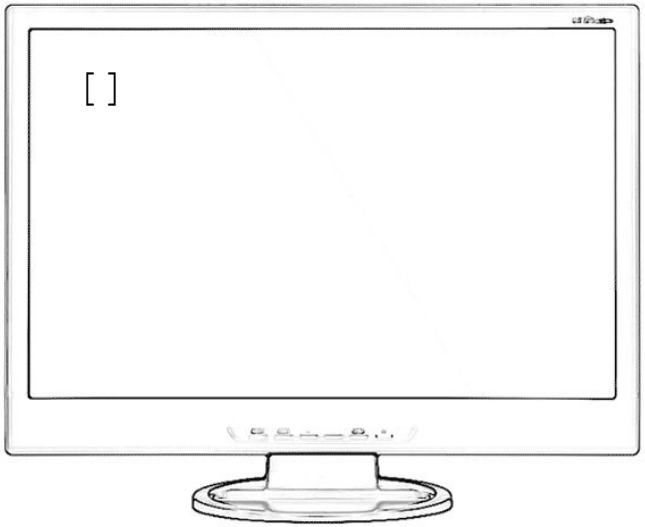
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
print(numbers)
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

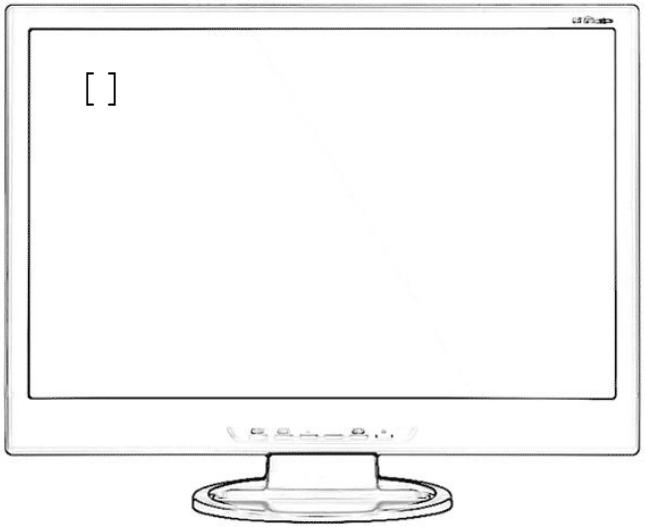
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

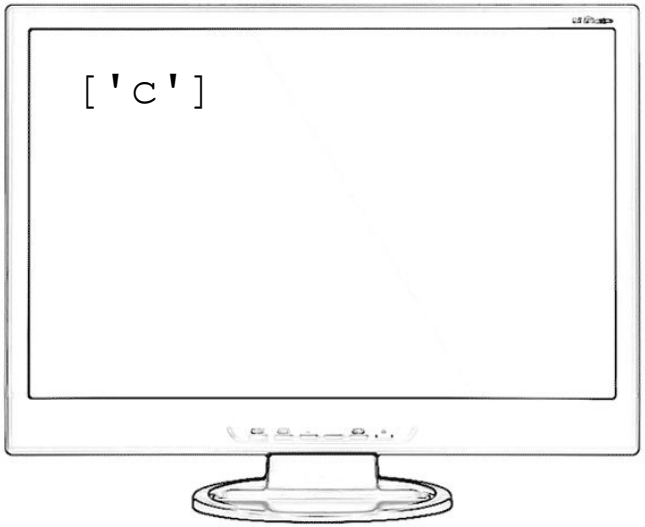
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

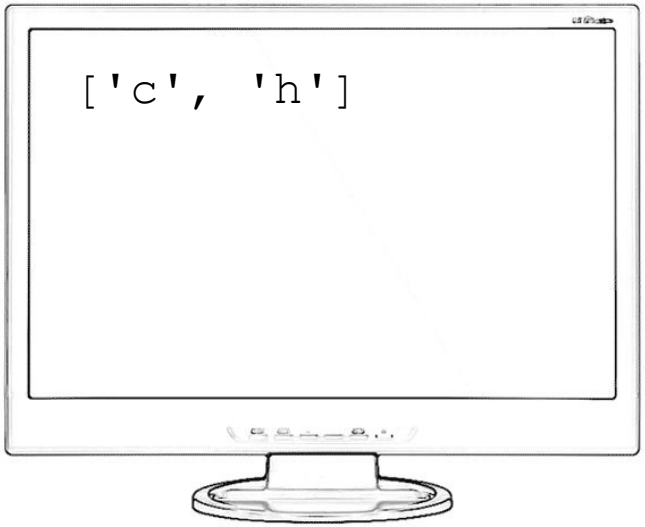
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

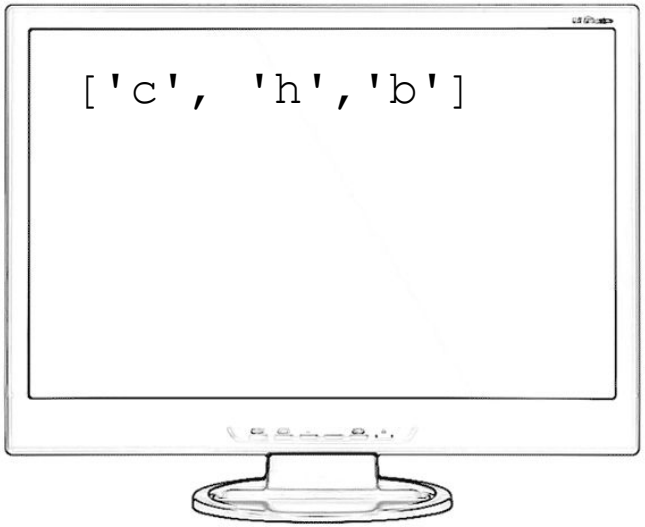
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# **Ответ:** Выведутся первые буквы слов

**Python:**

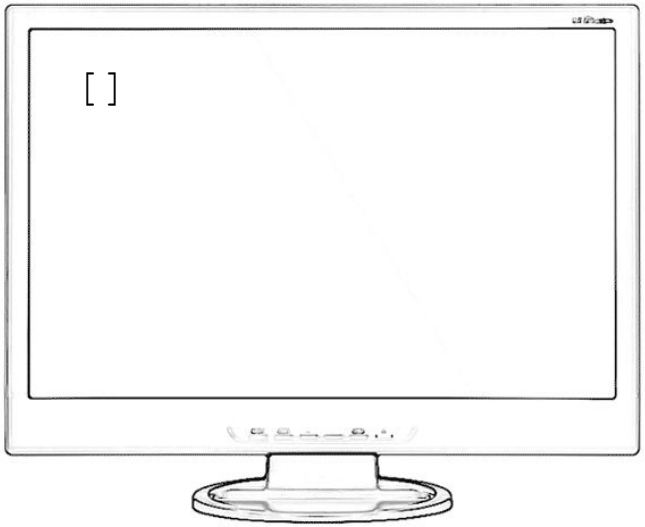
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# Как это читать?

**Python:**

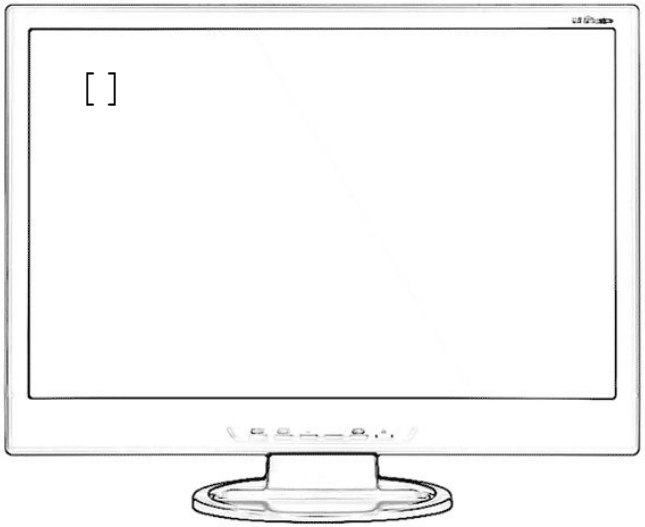
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# Как это читать?

**Python:**

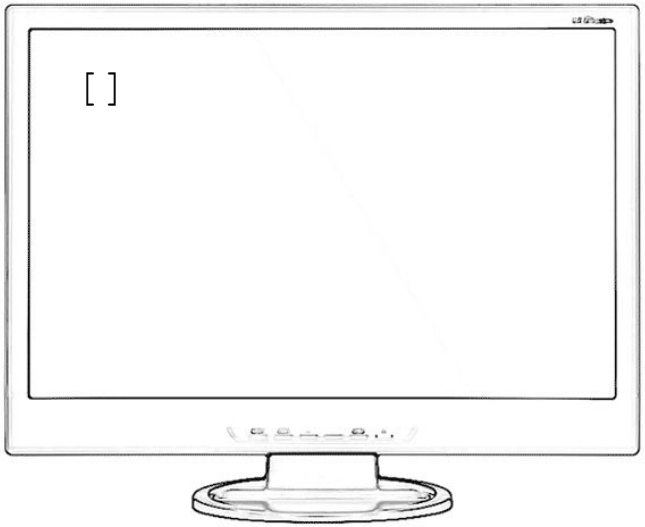
```
zoo = ['cat', 'hippo', 'bear']
numbers = [_x[0] for _x in zoo]
```



# Как это читать?

**Python:**

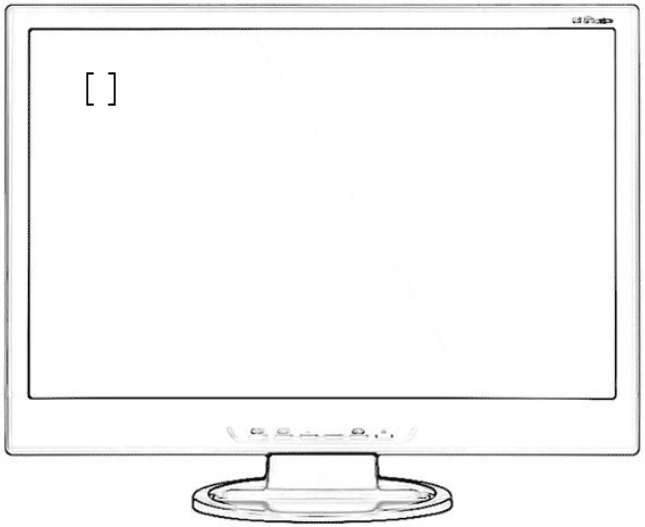
```
zoo = ['cat', 'hippo', 'bear']
numbers = ['c', 'h', 'b']
```



# Тренируемся

**Python:**

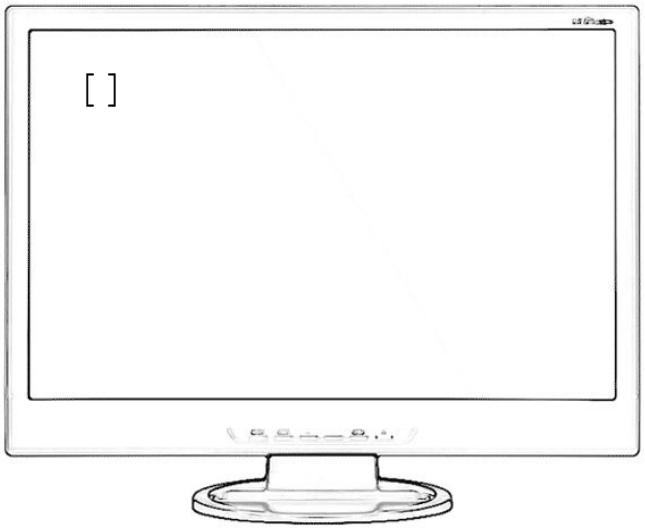
```
zoo = ['cat', 'hippo', 'bear']
numbers = [len(_x) for _x in zoo]
```



# Тренируемся

**Python:**

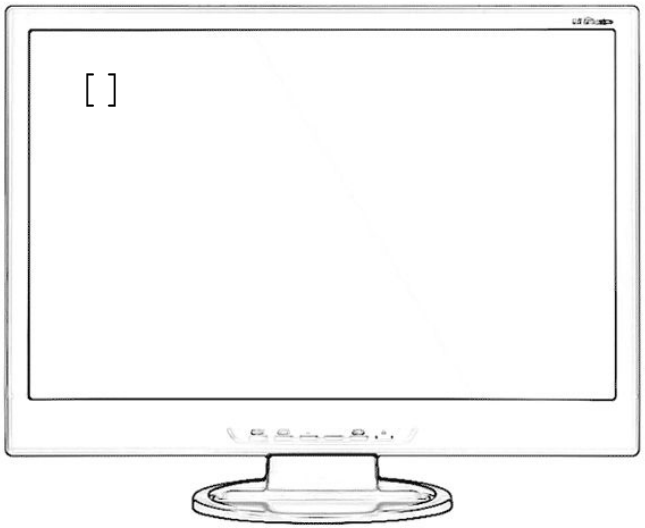
```
zoo = ['cat', 'hippo', 'bear']
numbers = [len(_x) for _x in zoo]
```



# Тренируемся

**Python:**

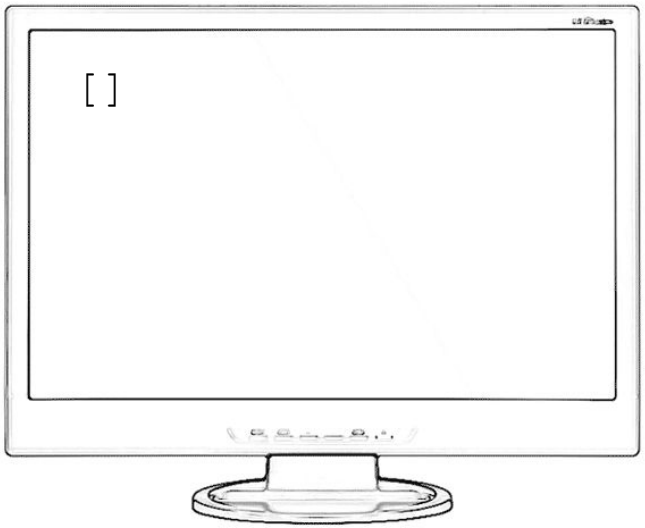
```
zoo = ['cat', 'hippo', 'bear']
numbers = [len(_x) for _x in zoo]
```



# Тренируемся

**Python:**

```
zoo = ['cat', 'hippo', 'bear']
numbers = [3, 5, 4]
```



```
zoo = ['cat', 'hippo', 'dog']
numbers = []
for i in range(len(zoo)):
    numbers.append(zoo[i][0])
```

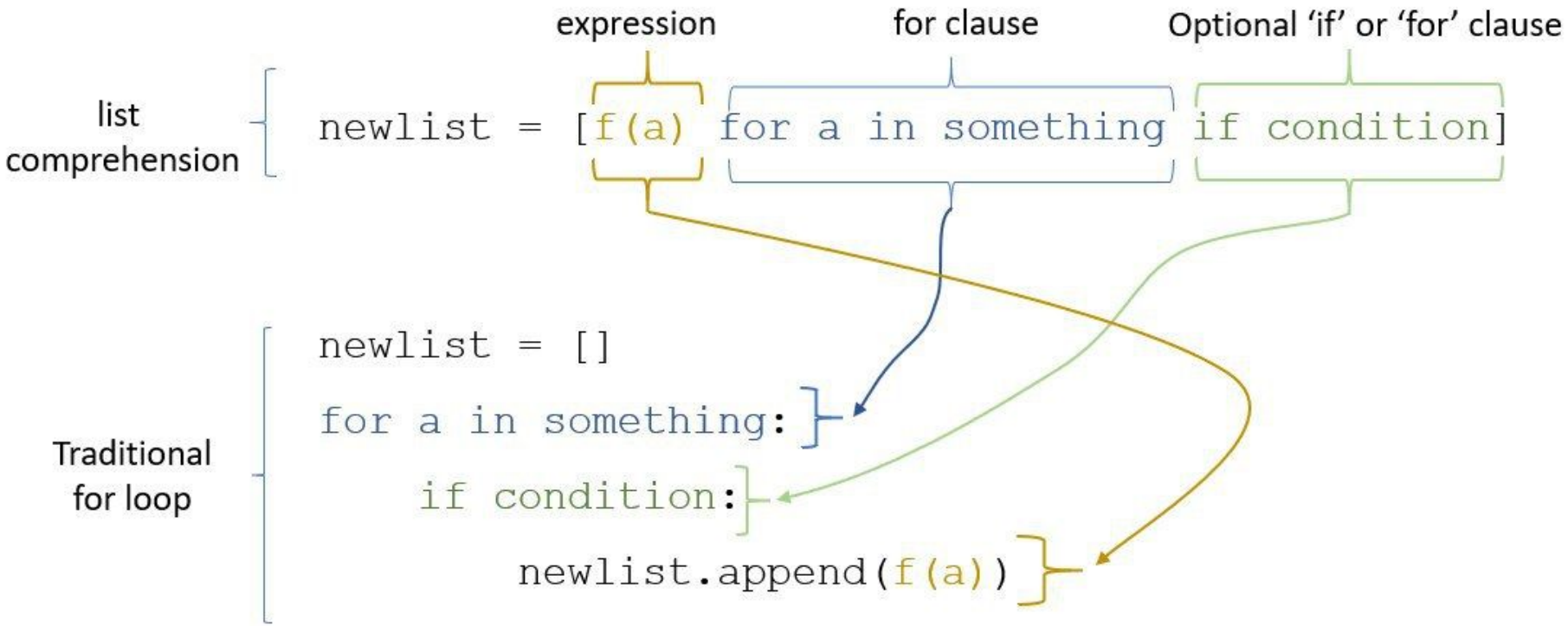


```
zoo = ['cat', 'hippo', 'dog']
numbers = []
for _x in zoo:
    numbers.append(_x[0])
```



```
zoo = ['cat', 'hippo', 'dog']
numbers = [_x[0] for _x in zoo]
```

# Generic Python list comprehension example



# КОЛЛЕКЦИИ

# КАК ЭТО ХРАНИТСЯ В ПАМЯТИ

# Как вы думаете, что будет на экране?

**№1**

```
s = 'cat'
s[2] = 'r'
print(s)
```

**№2**

```
l = ['c', 'a', 't']
l[2] = 'r'
print(''.join(l))
```

- Просто напишите свое предположение
- Пока без комментариев

**№3**

```
v = 2
u = v
u = 3
print(v)
```

**№4**

```
l = ['c', 'a', 't']
k = l
k[2] = 'r'
print(''.join(l))
```

**№5**

```
l = ['c', 'a', 't']
k = l
k = []
print(''.join(l))
```

# Join: Из списка в строку

№2

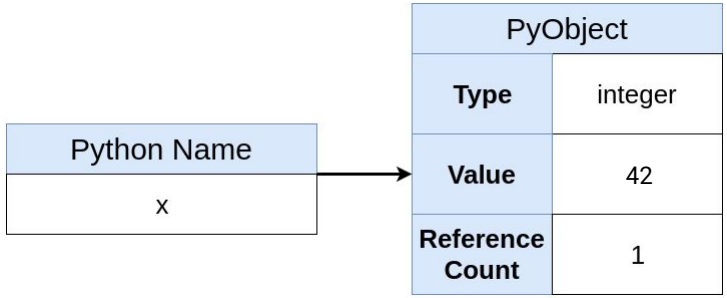
```
l = ['c', 'a', 't']
l[2] = 'r'
print(''.join(l))
```

['c', 'a', 't'] -> 'cat'

# Разбор

# Неизменяемые переменные в **Python**

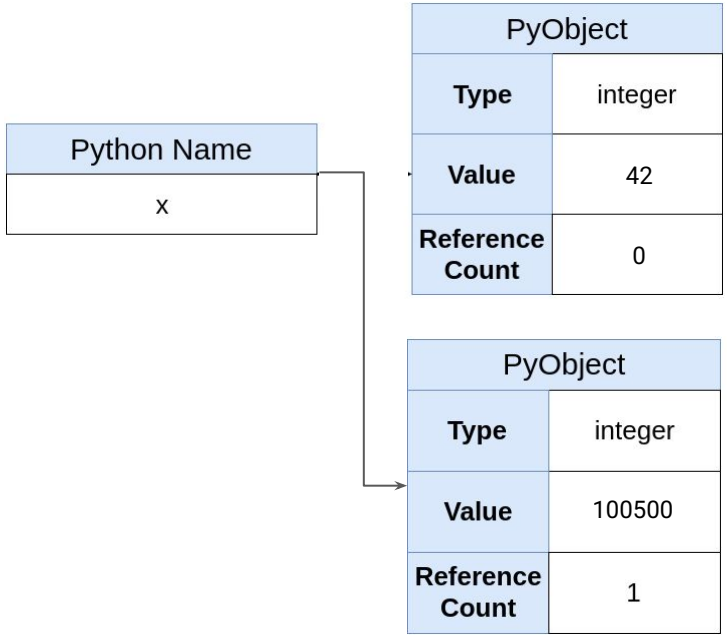
x = 42




# Неизменяемые переменные в **Python**

```
x = 42
```

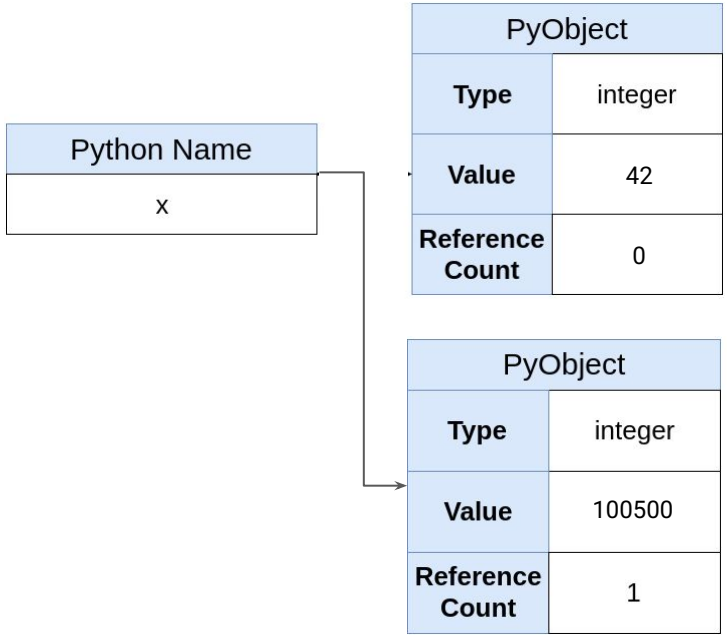
```
x = 100500
```



# Неизменяемые переменные в **Python**

```
x = 42
```

```
x = 100500
```

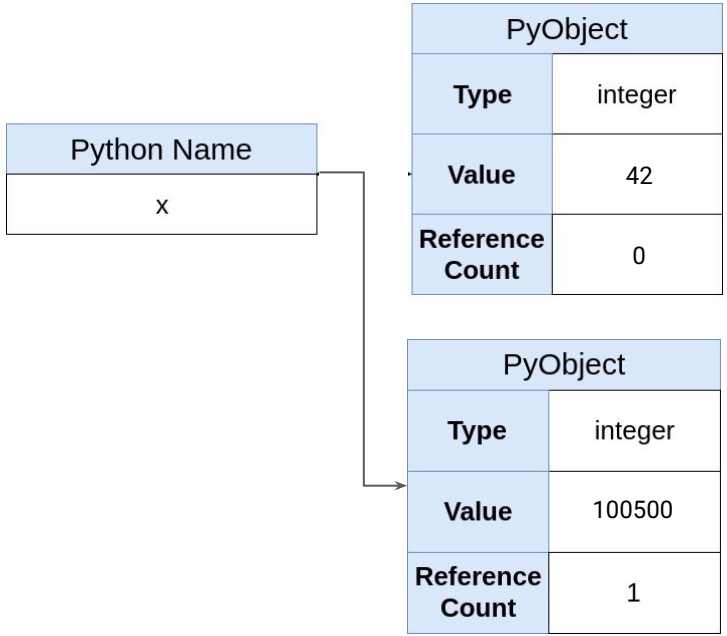



x указывает на ссылку на объект и не владеет областью памяти

# Неизменяемые переменные в **Python**

```
x = 42
```

```
x = 100500
```

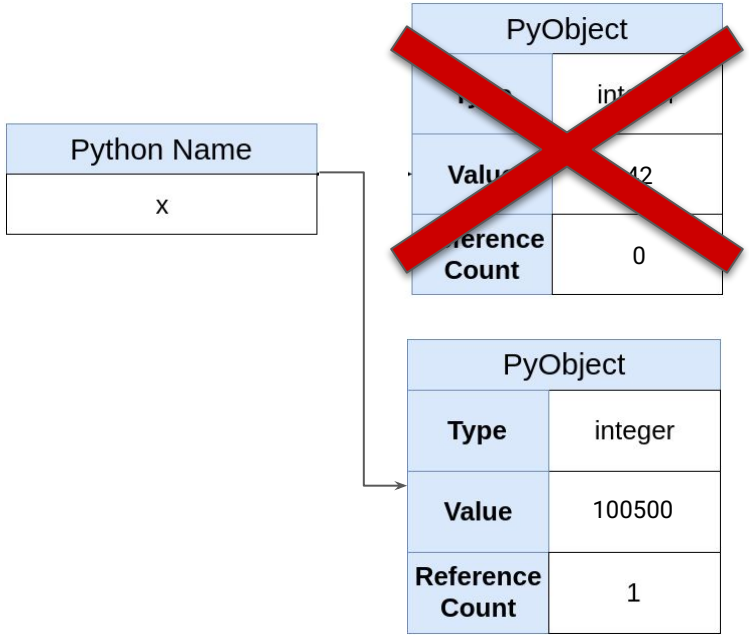



- Не переменные, а имена
- Не присваивание, а привязывание (binding)

# Неизменяемые переменные в **Python**

```
x = 42
```

```
x = 100500
```

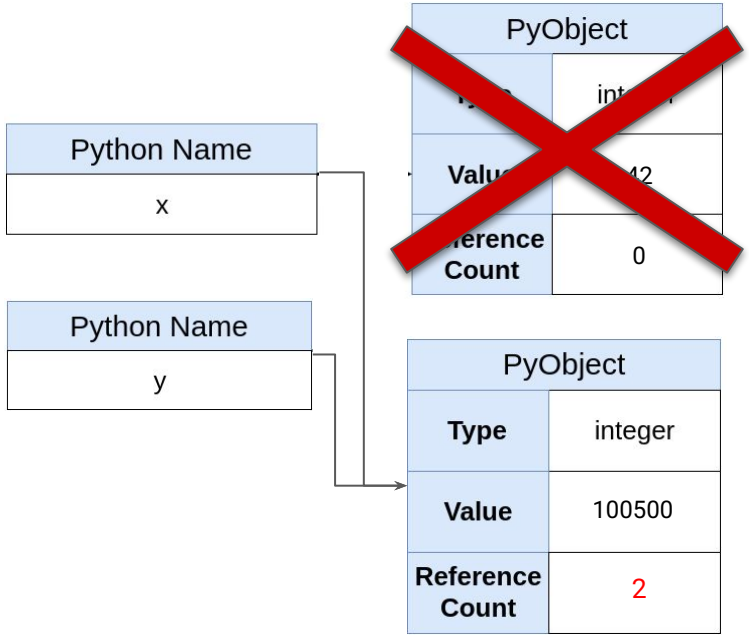



Будет удалена сборщиком мусора (Количество ссылок: 0)

# Неизменяемые переменные в **Python**

```
x = 42
```

```
x = 100500
y = x
```



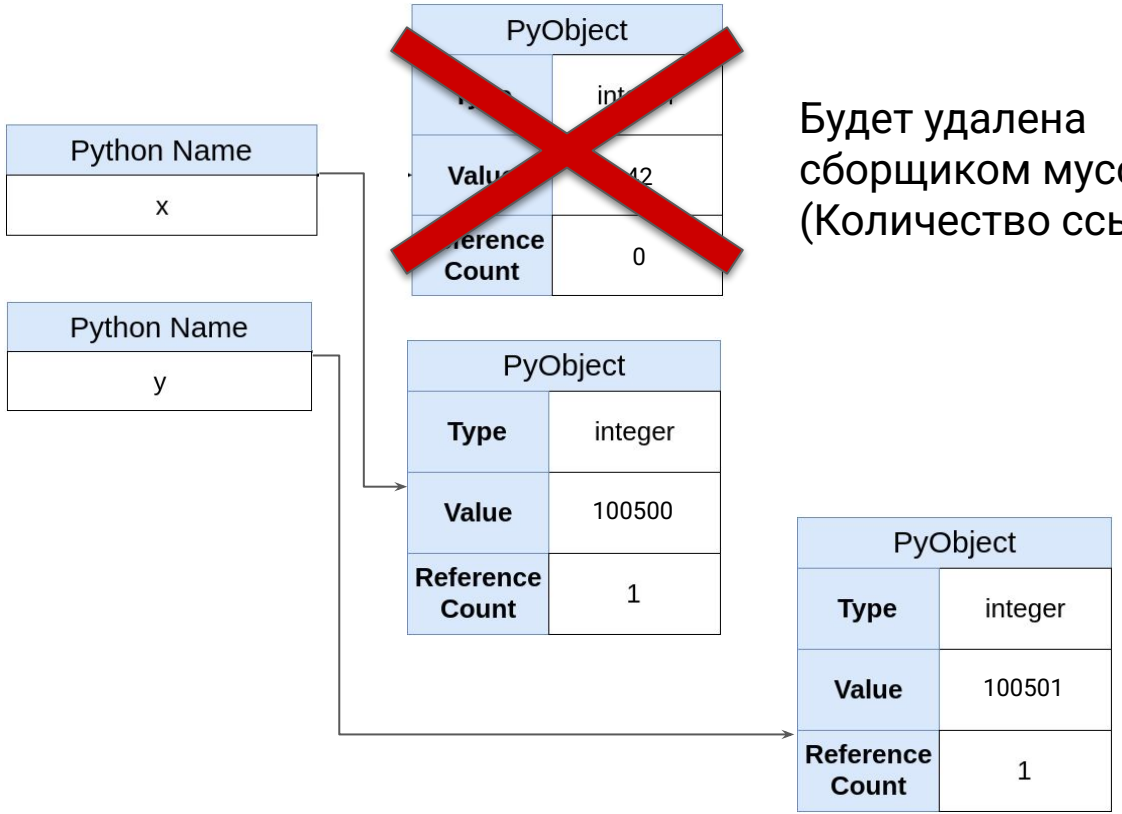

Будет удалена сборщиком мусора (Количество ссылок: 0)

# Неизменяемые переменные в **Python**

```
x = 42
```

```
x = 100500
y = x
```

```
y += 1
```




Будет удалена сборщиком мусора (Количество ссылок: 0)

# Типы объектов

Объекты бывают двух типов:

1. Неизменяемые объекты (не могут быть изменены):
   - int
   - float
   - bool
   - tuple

При попытке изменить эти объекты, мы бросим старую ячейку памяти, заведем новую и будем указывать на нее

2. Изменяемые объекты (могут быть изменены):
   - list
   - dict
   - set

- При попытке переприсвоения, мы также заведем новую ячейку памяти
- При попытке изменения (например, только одного значения списка), мы сможем просто изменить существующий объект

# Разбор

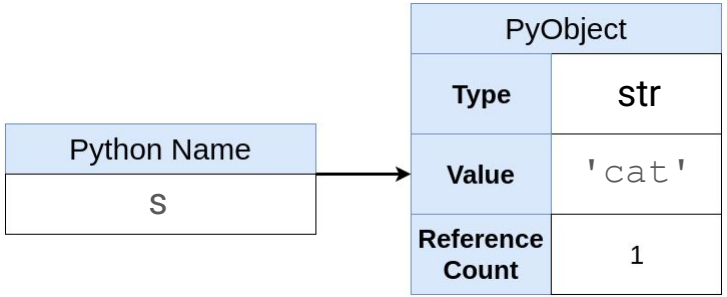
**№1**

```
s = 'cat'
s[2] = 'r'
print(s)
```

# Разбор

**№1**

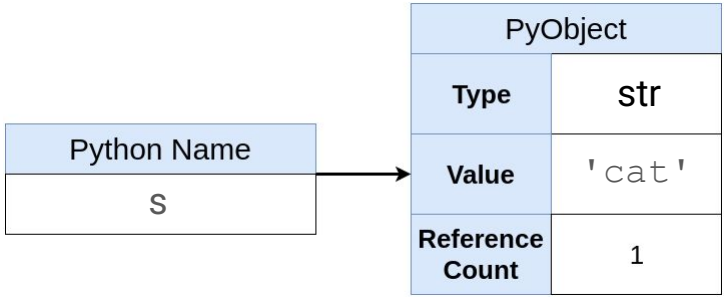
```
s = 'cat'
s[2] = 'r'
print(s)
```



# Разбор

**№1**

```
s = 'cat'
s[2] = 'r'
print(s)
```




**str** неизменяемый объект

Но и создать новый мы не можем, т.к. изменение частичное

**Правильный ответ:** Ошибка

# Разбор

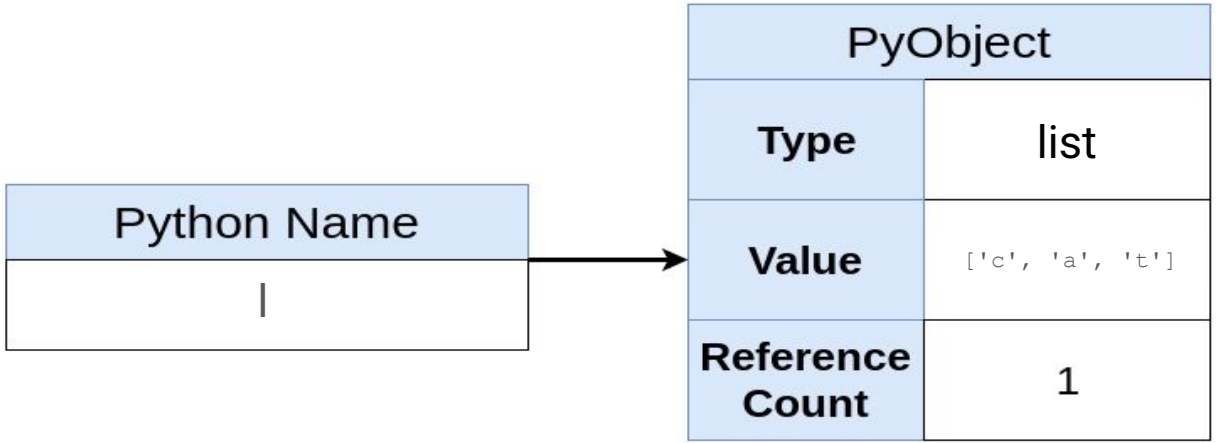
**№2**

```
l = ['c', 'a', 't']
l[2] = 'r'
print(''.join(l))
```

# Разбор

№2

```
l = ['c', 'a', 't']
l[2] = 'r'
print(''.join(l))
```



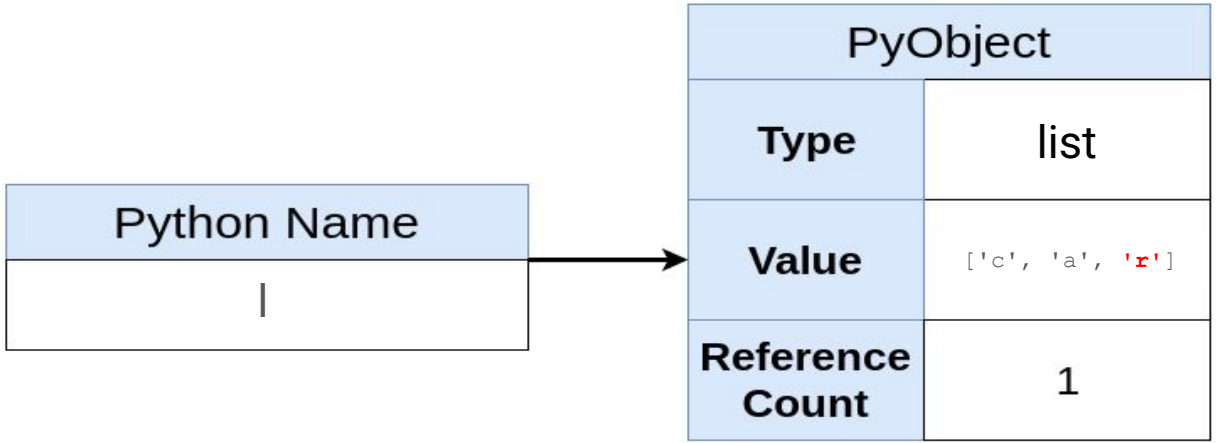
# Разбор

№2

```
l = ['c', 'a', 't']
l[2] = 'r'
print(''.join(l))
```

List изменяемый тип.

**Ответ:** “car”



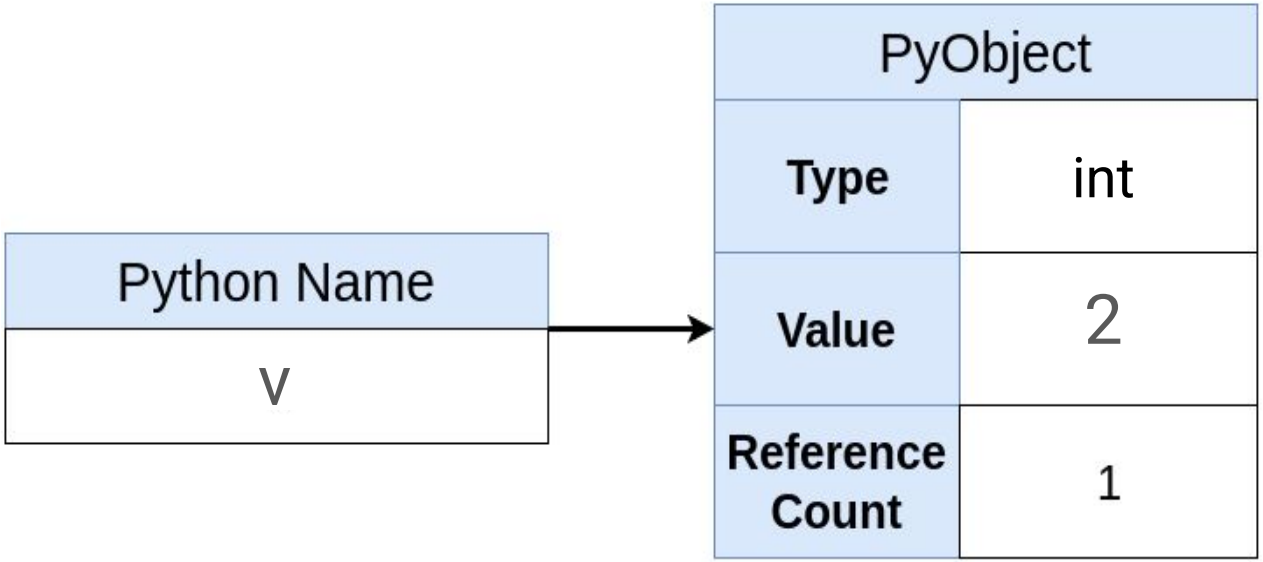
# Разбор

**№3**

```
v = 2
u = v
u = 3
print(v)
```

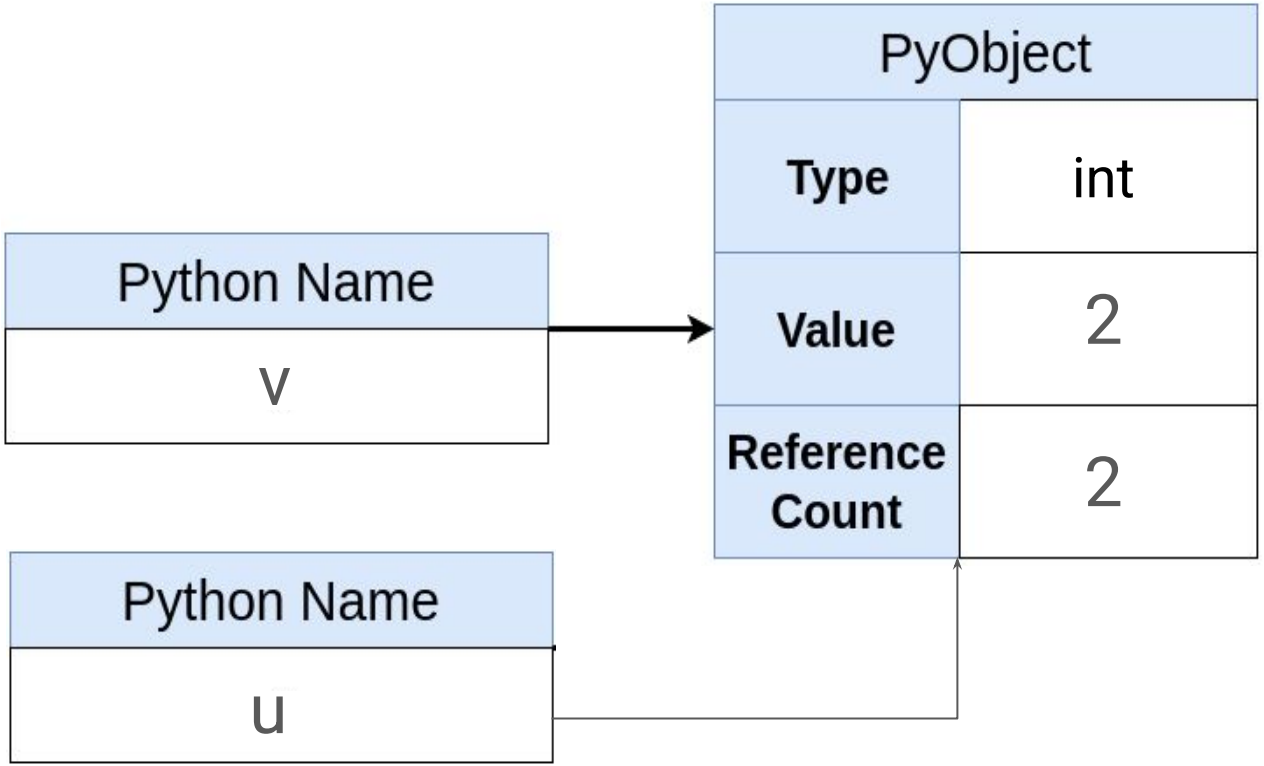
# Разбор

```
№3
v = 2
u = v
u = 3
print(v)
```



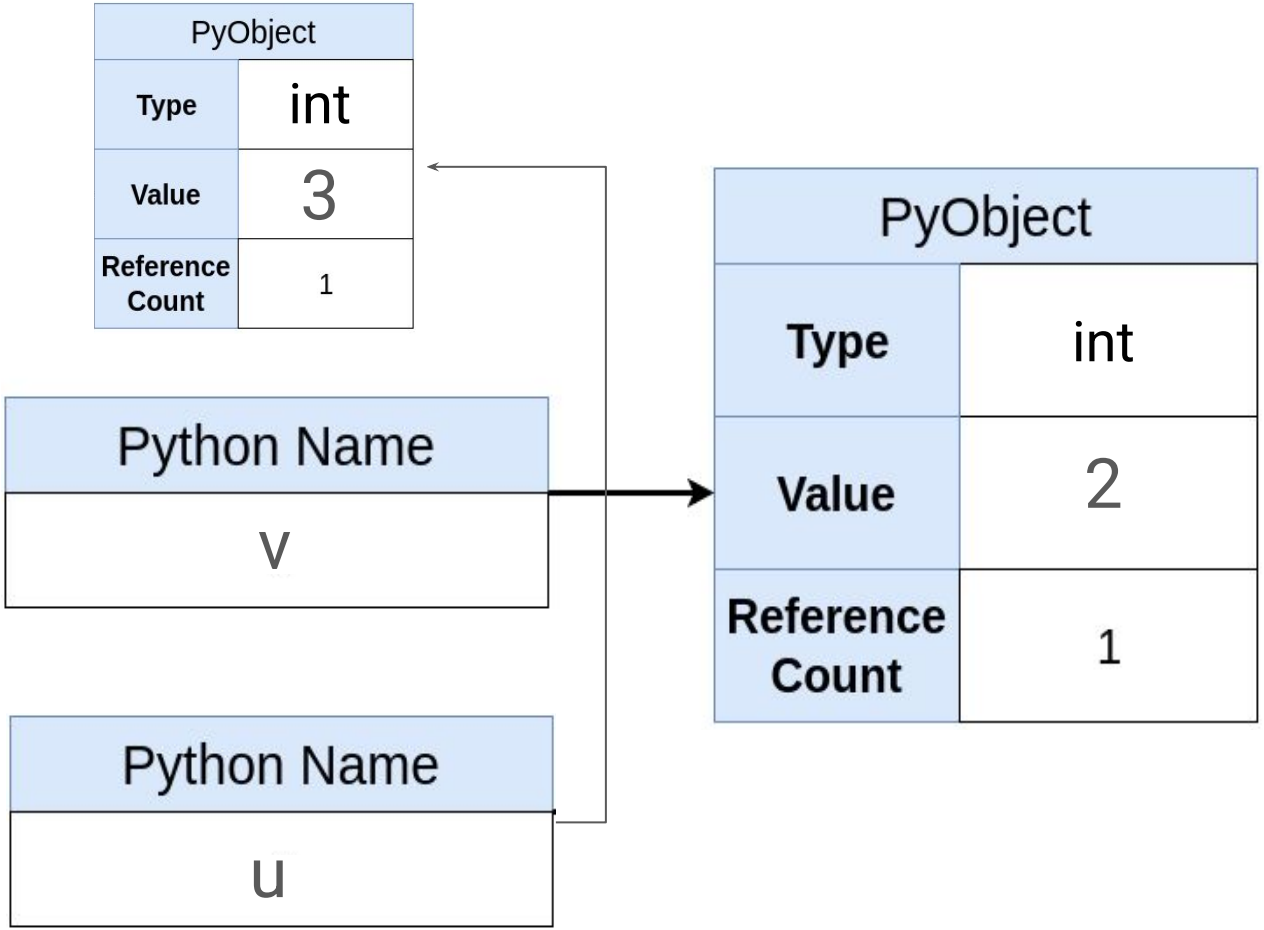
# Разбор

```
№3
v = 2
u = v
u = 3
print(v)
```



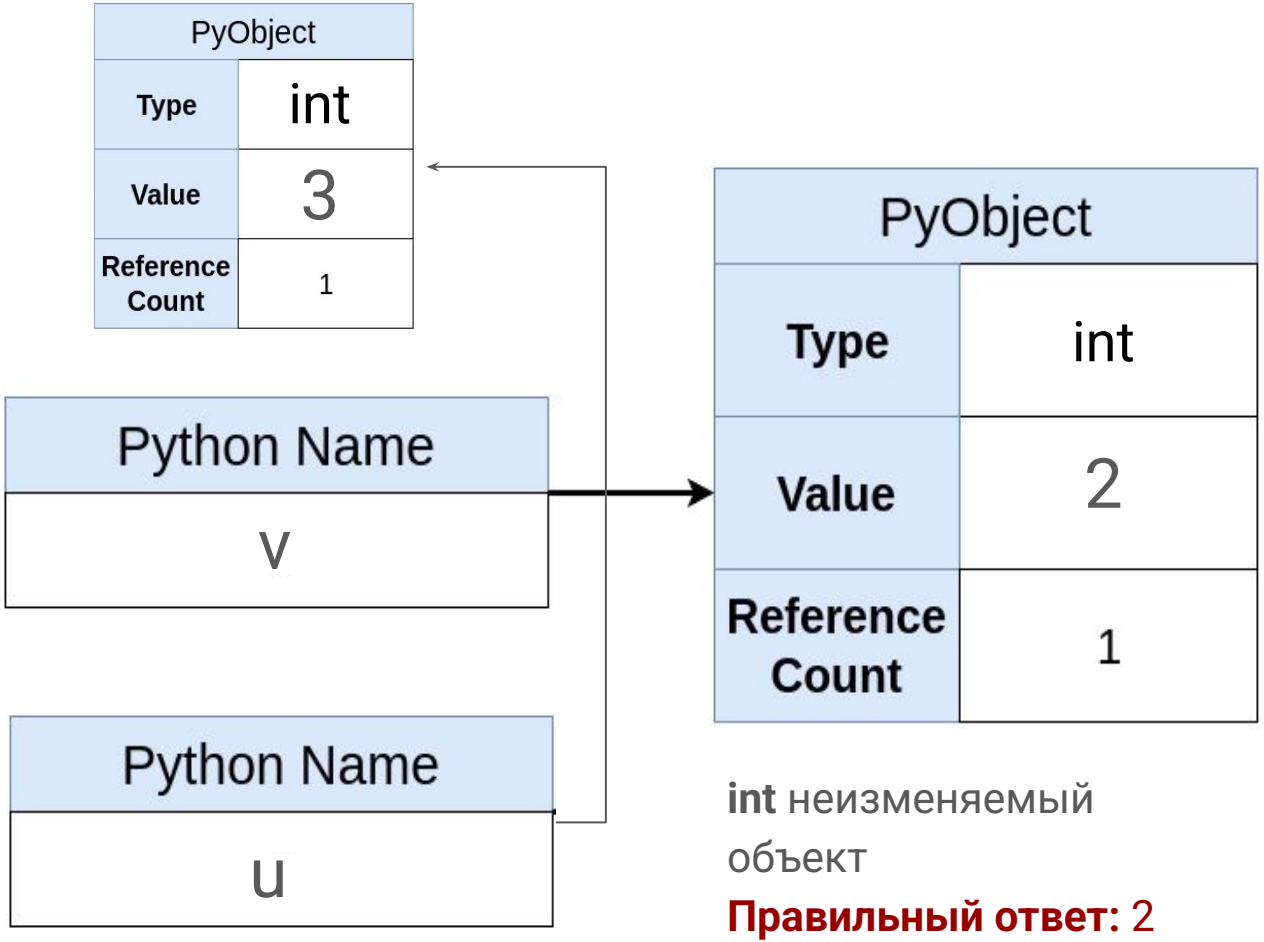
# Разбор

```
№3
v = 2
u = v
u = 3
print(v)
```



# Разбор

```
№3
v = 2
u = v
u = 3
print(v)
```




**int** неизменяемый объект

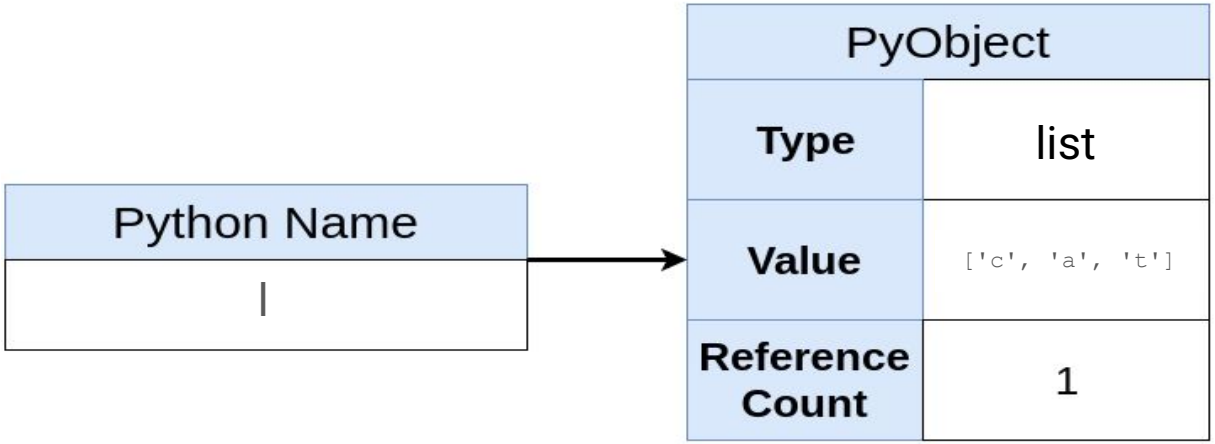
**Правильный ответ:** 2

# Разбор

**№4**

```
l = ['c', 'a', 't']
k = l
k[2] = 'r'
print(''.join(l))
```

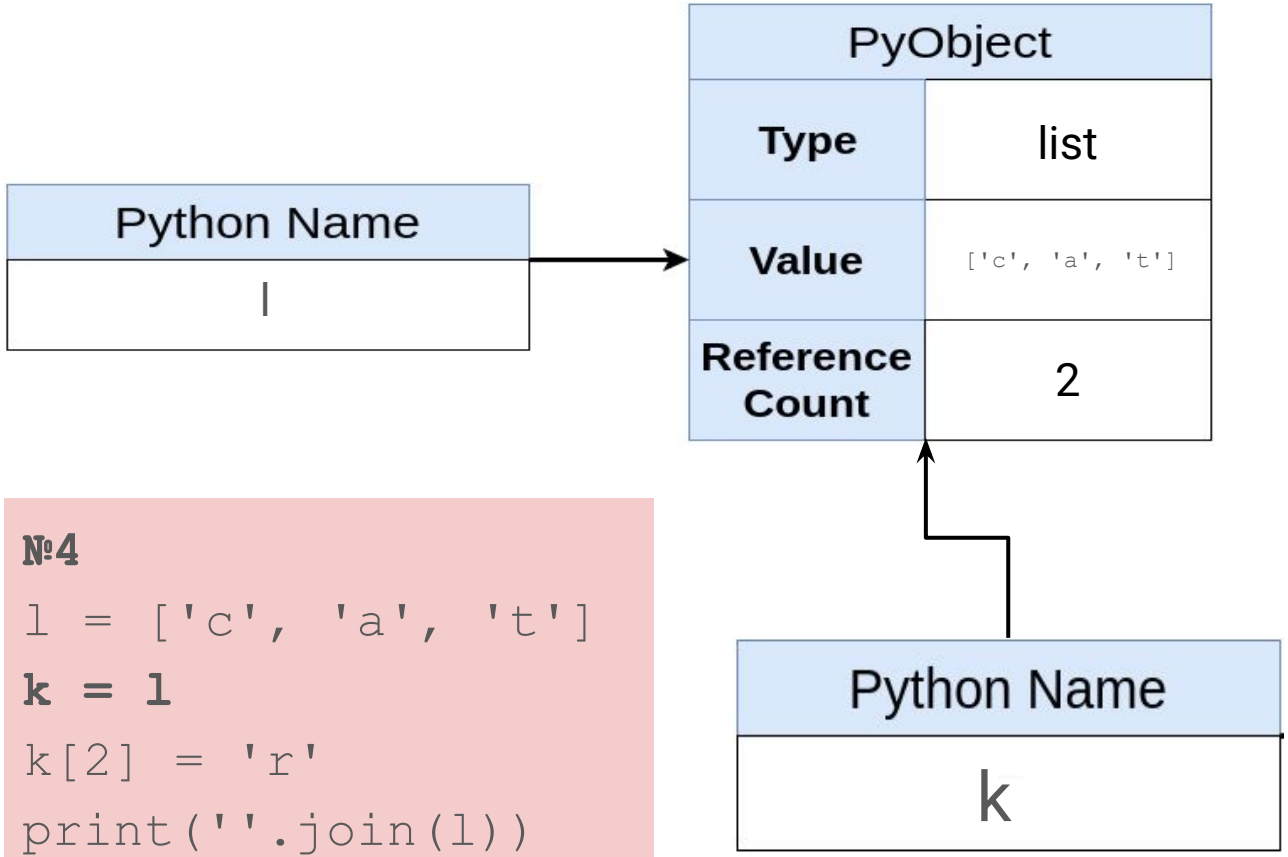
# Разбор




**№4**

```
l = ['c', 'a', 't']
k = l
k[2] = 'r'
print(''.join(l))
```

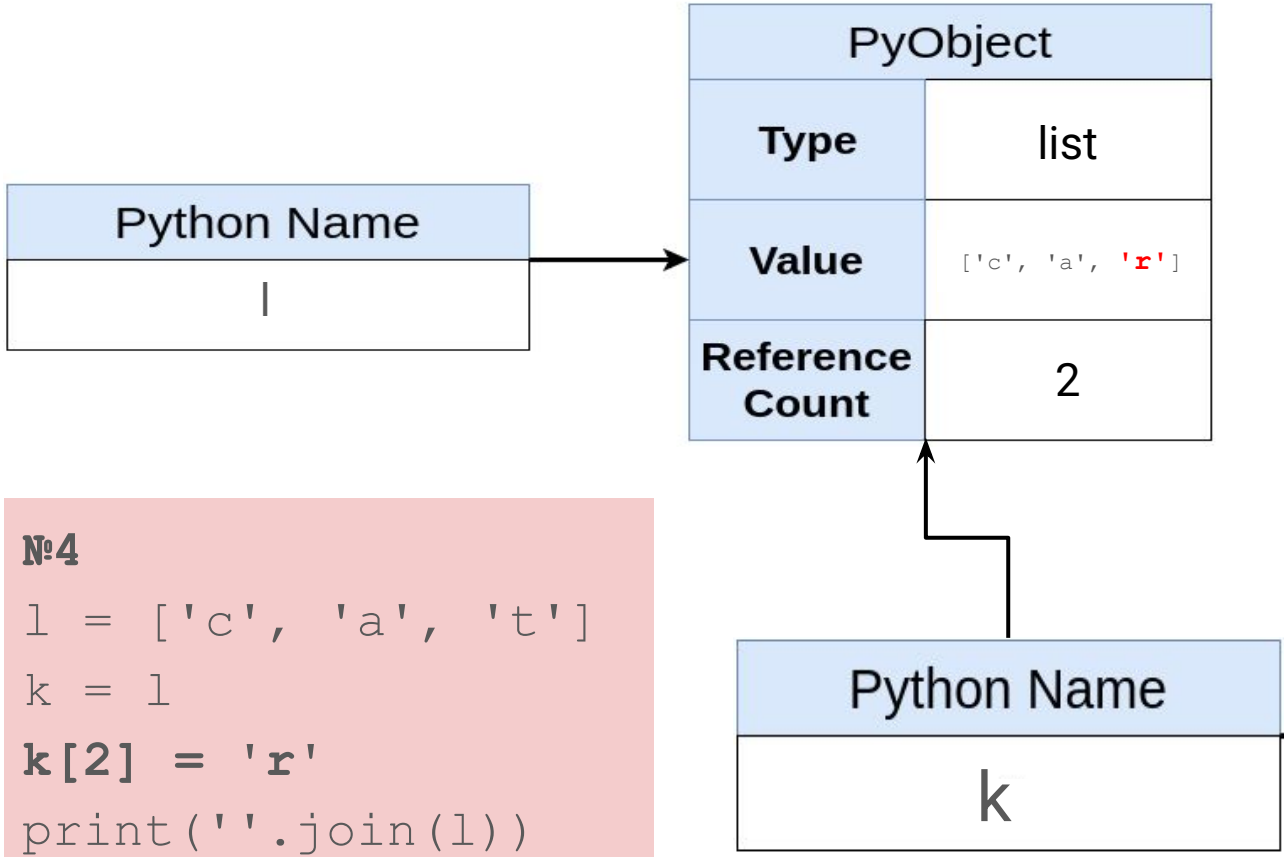
# Разбор




**№4**

```
l = ['c', 'a', 't']
k = l
k[2] = 'r'
print(''.join(l))
```

# Разбор



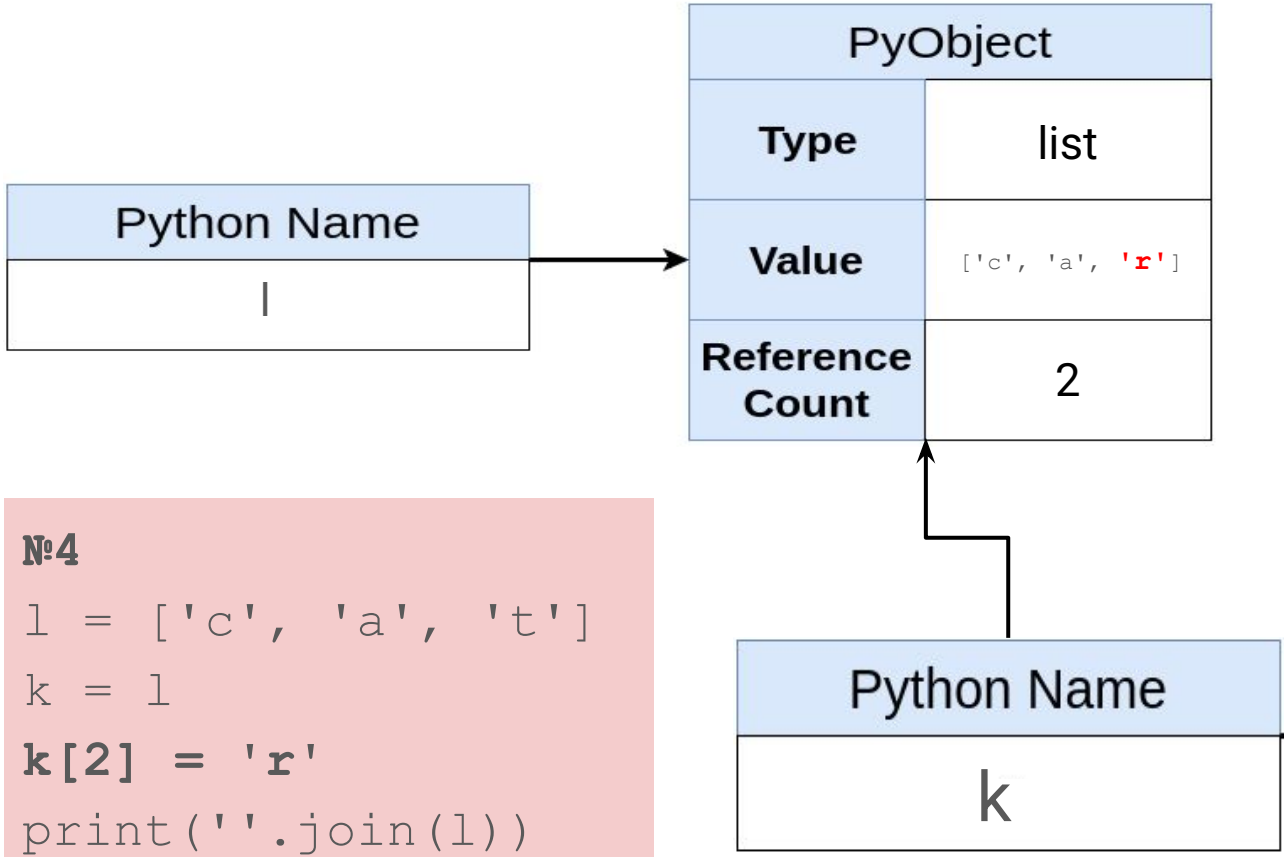

**№4**

```
l = ['c', 'a', 't']
k = l
k[2] = 'r'
print(''.join(l))
```

# Разбор

List изменяемый тип.

**Ответ:** “car”




**№4**

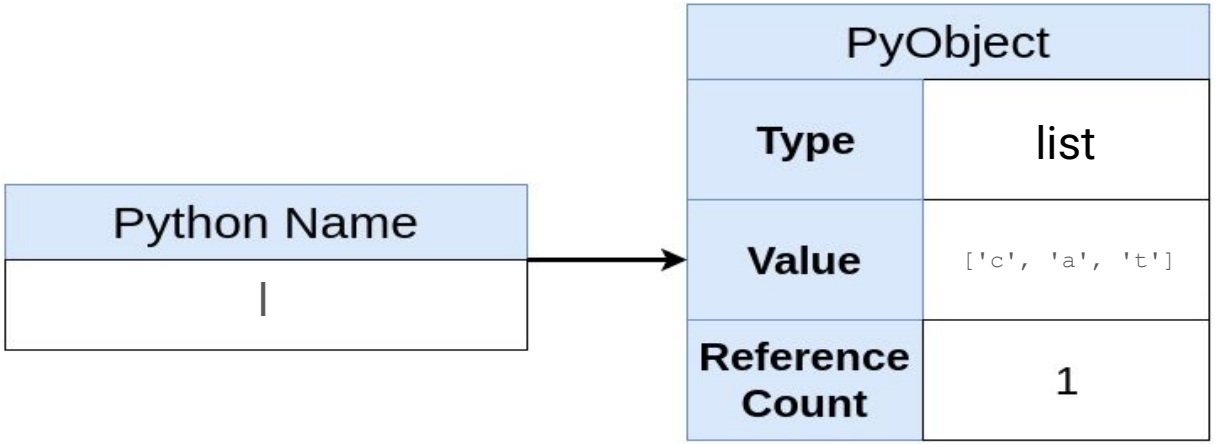
```
l = ['c', 'a', 't']
k = l
k[2] = 'r'
print(''.join(l))
```

# Разбор

№5

```
l = ['c', 'a', 't']
k = l
k = []
print(''.join(l))
```

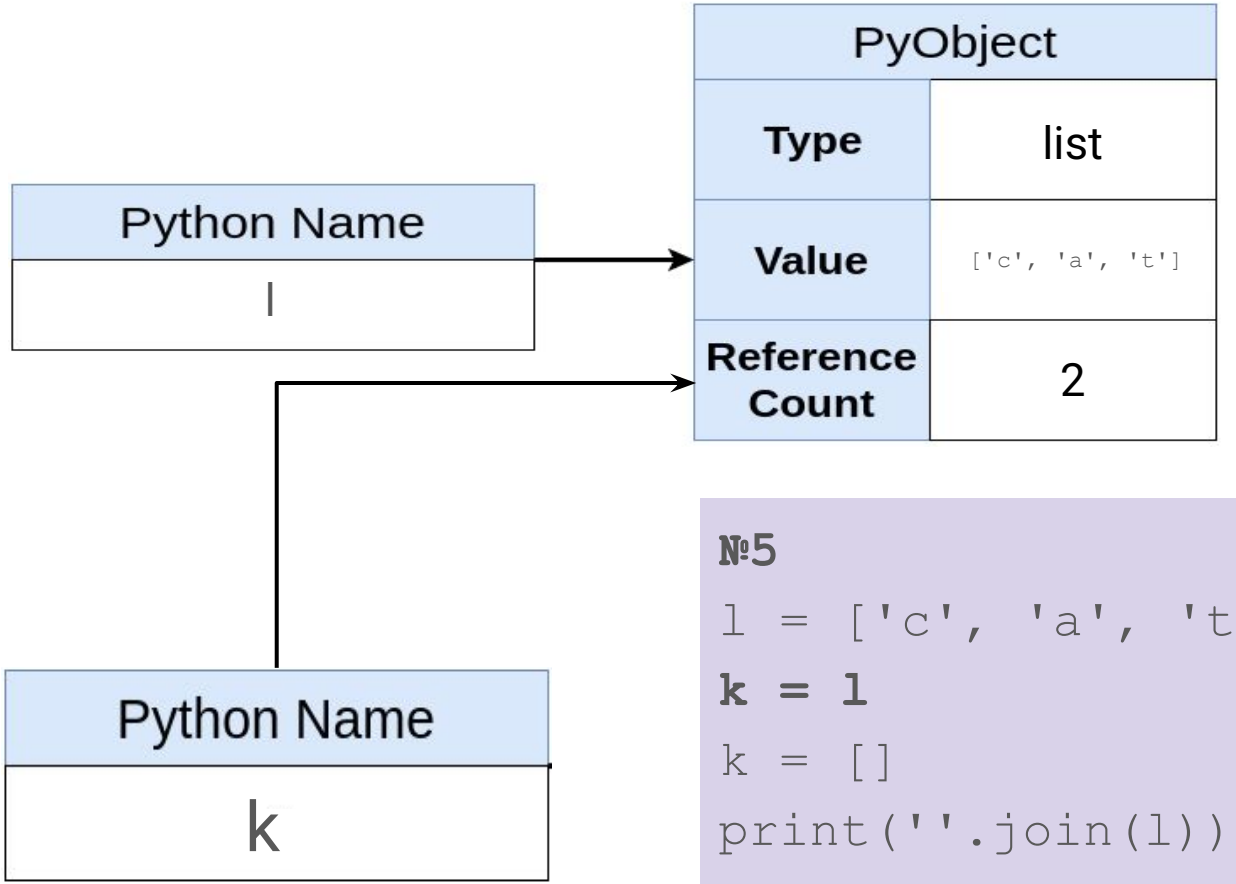
# Разбор




№5

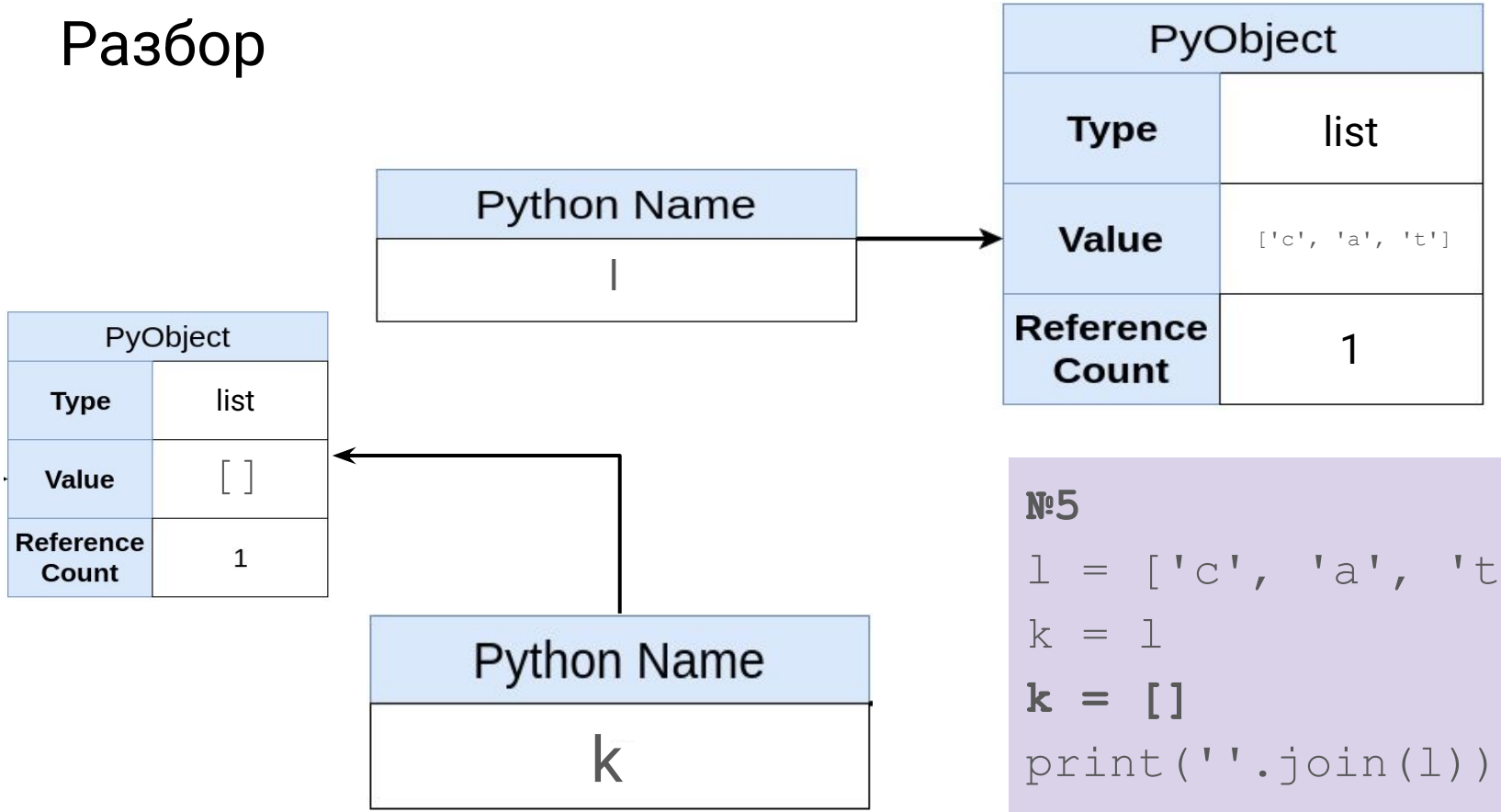
```
l = ['c', 'a', 't']
k = l
k = []
print(''.join(l))
```

# Разбор




```
№5
l = ['c', 'a', 't']
k = l
k = []
print(''.join(l))
```

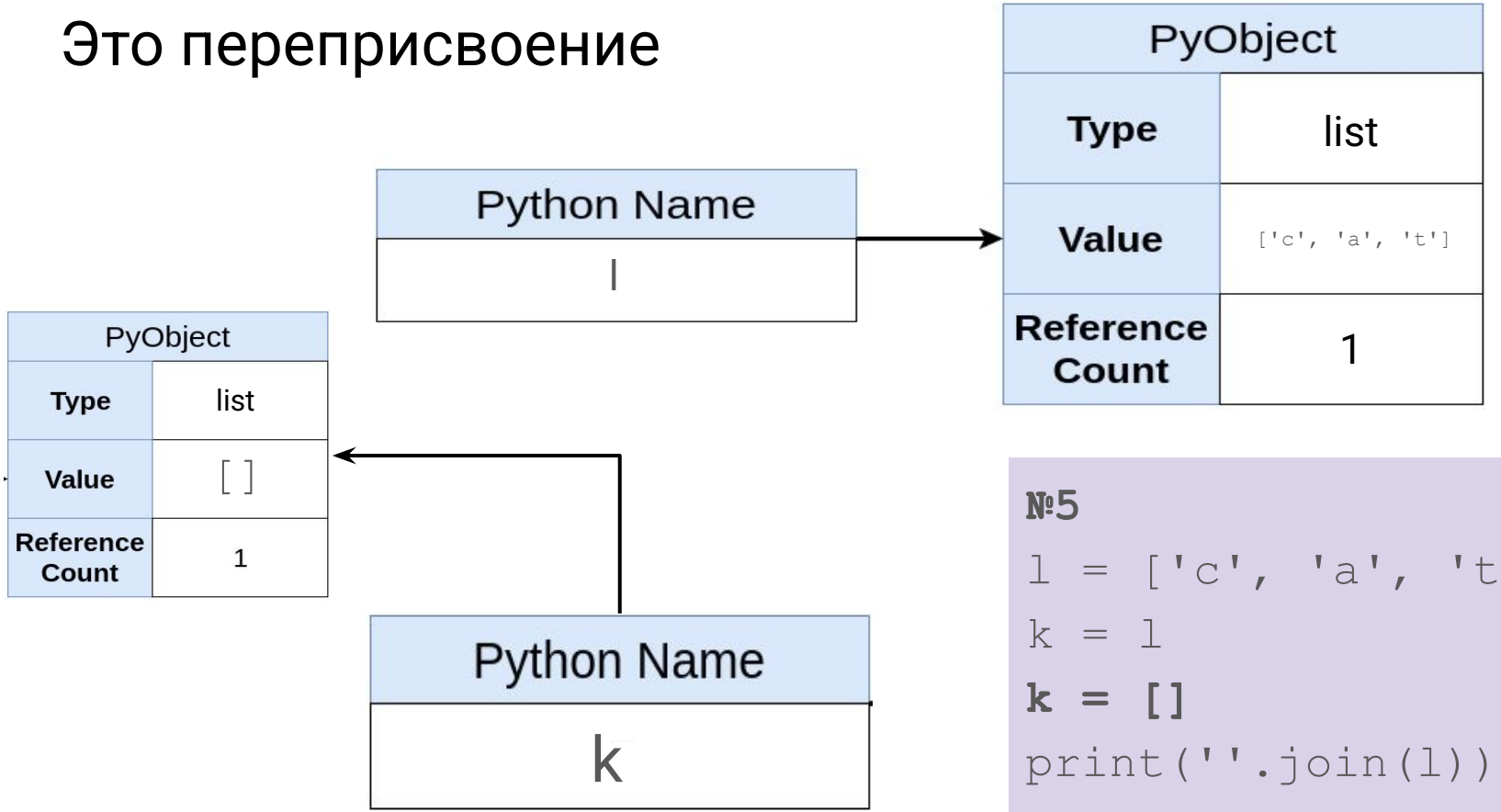
# Разбор




№5

```
l = ['c', 'a', 't']
k = l
k = []
print(''.join(l))
```

# Это переприсвоение




**№5**

```
l = ['c', 'a', 't']
k = l
k = []
print(''.join(l))
```

# Кортеж
# **Tuple**

# Типы объектов

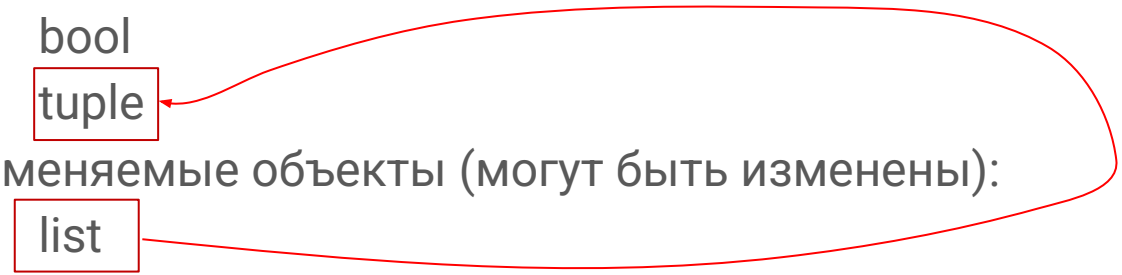
Объекты бывают двух типов:

1. Неизменяемые объекты (не могут быть изменены):
    - int
    - float
    - bool
    - tuple
2. Изменяемые объекты (могут быть изменены):
    - list
    - dict
    - set

# Типы объектов

Объекты бывают двух типов:

1. Неизменяемые объекты (не могут быть изменены):
   - int
   - float
   - bool
   - tuple
2. Изменяемые объекты (могут быть изменены):
   - list
   - dict
   - set



hashable

immutable

| | List | Tuple | Dict | Set |
|---|---|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | | | |
| Инициализация | `elements = [200, 300]`<br>`elements =`<br>`"2+3".split("2+3")` | `pair = (200, 300)`<br>`pair = tuple([200, 300])` | | |
| Обращение к элементу | `v = elements[1]`<br>`v = elements[1:3]` | `v = pair[0]` | | |
| Добавление | `elements.append(400)` | ✗ | | |
| Расширение | `elements += [500, 600]` | ✗ | | |
| Изменение | `elements[7] = "bob"`<br>`elements[1:2] = [0, 0]` | ✗ | | |
| Проверка наличия | `if 200 in elements:` | `if 200 in pair:` | | |
| Comprehension | `e = [_a for _a in elements]` | ✗ | | |

# Словарь
# **Dict**

# Задача №1

Вводят 4 числа  (0, …, 9). Посчитать сколько раз встретилось каждое

**Пример входных данных:**
9
1
0
9

**Выходные данные:**
1 1 0 0 0 0 0 0 0 2

# Задача №1

Если бы нам надо было посчитать только 0

```
for i in range(4):
  value = int(input())
  …
```

# Задача №1

Если бы нам надо было посчитать только 0

```
counter = 0

for i in range(4):
  value = int(input())
  …


print(counter)
```

# Задача №1

Если бы нам надо было посчитать только 0

```
counter = 0

for i in range(4):
  value = int(input())
  if value == 0:
    counter = counter + 1

print(counter)
```

# Задача №1

А теперь все числа:

```
counters = []

for i in range(4):
  value = int(input())
  if value == 0:
    counter = counter + 1

print(counter)
```

# Задача №1

А теперь все числа:

```
counters = [0 for _ in range(10)]

for i in range(4):
  value = int(input())
  if value == 0:
    counter = counter + 1

print(counter)
```

# Задача №1

А теперь все числа:

```
counters = [0 for _ in range(10)]

for i in range(4):
  value = int(input())
  counters[value] = counters[value] + 1

print(counters)
```

# Словари
# **Dict**

# Задача №2

Посчитать количество каждого слова.

**Пример входных данных:**
la la tancuyut zvezdy

**Выходные данные:**
la 2
tancuyut 1
zvezdy 1

# Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(foo[1])

x = foo[-1]
```

Обращение

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(foo[1])
x = foo[-1]
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(bar["petrov"])
x = bar["trump"]
```

Присваивание

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo[2] = 555
foo[5] = 0
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

Присваивание

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo[2] = 555
foo[5] = 0
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

Присваивание

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo[2] = 555
foo[5] = 0
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
bar["trump"] = 555
bar["someone"] = 0
```

Присваивание

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo[2] = 555
foo[5] = 0
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump | someone |
|---|---|---|---|---|
| 300 | 400 | 100 | 900 | 0 |

```
bar["trump"] = 555
bar["someone"] = 0
```

Задание начальных значений

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo = [300, 400, 100, 900]
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
bar = {
    'ivanov': 300,
    'petrov': 400,
    'sidorov': 100,
    'trump': 900,
}
```

Задание начальных значений

# Список

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo = []
foo = list()
```

# Словарь

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
bar = {}
bar = dict()
```

# Список

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
foo = list()
foo.append(300)
```

# Словарь

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
bar = dict()
bar['ivanov'] = 200
```

| | List | Dict |
|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | `mapping = {}`<br>`mapping = dict()` |
| Инициализация | `elements = [200, 200, 200]` | `mapping = {`<br>`  "Аркадьич": 200,`<br>`  "Борисыч": 200`<br>`}` |
| Задание значения | `elements[7] = "bob"` | `mapping[7] = "bob"`<br>`mapping["bob"] = 7` |
| Проверка наличия | `if 200 in elements:` | `if 200 in mapping.values():`<br>`if "bob" in mapping.keys():` |
| Comprehension | `e = [_a for _a in elements]` | `m = {_k: _v for _k, _v in mapping.items()}` |

Обращение

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(foo[1])
x = foo[-1]
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(bar["petrov"])
x = bar["someone"]
```

Обращение

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(foo[1])
x = foo[-1]
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(bar["petrov"])
x = bar["someone"]
```

Обращение

## Список **foo**

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(foo[1])

x = foo[-1]
```

## Словарь **bar**

| ivanov | petrov | sidorov | trump |
|---|---|---|---|
| 300 | 400 | 100 | 900 |

```
print(bar["petrov"])

x = bar.get("someone", -1)
```

# Задача №2

А теперь все ~~числа~~ слова:

```
counters = [0 for _ in range(10)]

for i in range(4):
  value = int(input())
  counters[value] = counters[value] + 1

print(counters)
```

# Задача №2

А теперь все ~~числа~~ слова:

```
counters = dict()

for i in range(4):
  value = input()
  counters[value] = counters[value] + 1

print(counters)
```

# Задача №2

А теперь все ~~числа~~ слова:

```
counters = dict()

for i in range(4):
  value = input()
  counters[value] = counters.get(value, 0) + 1

print(counters)
```

hashable

immutable

| | List | Tuple | Dict | Set |
|---|---|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | | `mapping = {}`<br>`mapping = dict()` | |
| Инициализация | `elements = [200, 300]`<br>`elements = "2+3".split("2+3")` | `pair = (200, 300)`<br>`pair = tuple([200, 300])` | `mapping = {`<br>`  "Аркадьич": 200,`<br>`  "Борисыч": 300`<br>`}` | |
| Обращение к элементу | `v = elements[1]`<br>`v = elements[1:3]` | `v = pair[0]` | `v = elements["Аркадьич"]`<br>`v = elements.get("Ким", -1)` | |
| Добавление | `elements.append(400)` | ✗ | `v["Владимирович"] = 400` | |
| Расширение | `elements += [500, 600]` | ✗ | `elements.update(other_dict)` | |
| Изменение | `elements[7] = "bob"`<br>`elements[1:2] = [0, 0]` | ✗ | `mapping[7] = "bob"`<br>`mapping["bob"] = 7` | |
| Проверка наличия | `if 200 in elements:` | `if 200 in pair:` | `if 200 in mapping.values():`<br>`if "bob" in mapping.keys():` | |
| Comprehension | `e = [_a for _a in elements]` | ✗ | `m = {_k: _v for _k, _v in mapping.items()}` | |

# Множества
# **set**

# set

**Задача:** посчитать, сколько в списке уникальных чисел?

```
values = [9, 2, 1, 2, 7, 1, 8, 8, 8, 9, 3, 5, 9, 7, 4, 8, 9,
3, 1, 2, 5, 6, 8, 9, 9, 1, 9, 8, 3, 4, 9, 2, 8, 6, 6, 7, 4,
4, 9, 4, 7, 7, 6, 6, 6, 8, 6, 2, 9, 9, 9, 8, 7, 8, 2, 3, 1,
7, 2, 3, 3, 5, 1, 5, 3, 4, 1, 7, 7, 5, 7, 7, 5, 9, 5, 1, 5,
6, 8, 2, 9, 8, 7, 3, 8, 6, 6, 8, 5, 8, 4, 5, 8, 4, 9, 4, 6,
7, 8, 5, 3, 1, 9, 2, 1, 6, 6, 6, 2, 1, 5, 2, 1, 9, 1, 9, 6,
7, 4, 9, 8, 2, 3, 6, 5, 1, 3, 4, 5, 2, 4, 4, 1, 4, 2, 8, 6,
3, 1, 8, 6, 2, 2, 7, 9, 5, 1, 6, 9, 7, 3, 3, 1, 7, 8, 9, 8,
8, 9, 4, 9, 6, 2, 2, 6, 7, 8, 4, 6, 7, 8, 5, 9, 8, 9, 8, 7,
9, 6, 8, 6, 5, 9, 5, 9, 1, 8, 3, 6, 3, 8, 5, 6, 8, 6, 1, 9,
9, 3, 5]
```

# set

**Задача:** посчитать, сколько в списке уникальных чисел?

```
values = [9, 2, 1, 2, 7, 1, 8, 8, 8, 9, 3, 5, 9, 7, 4, 8, 9,
3, 1, 2, 5, 6, 8, 9, 9, 1, 9, 8, 3, 4, 9, 2, 8, 6, 6, 7, 4,
4, 9, 4, 7, 7, 6, 6, 6, 8, 6, 2, 9, 9, 9, 8, 7, 8, 2, 3, 1,
7, 2, 3, 3, 5, 1, 5, 3, 4, 1, 7, 7, 5, 7, 7, 5, 9, 5, 1, 5,
6, 8, 2, 9, 8, 7, 3, 8, 6, 6, 8, 5, 8, 4, 5, 8, 4, 9, 4, 6,
7, 8, 5, 3, 1, 9, 2, 1, 6, 6, 6, 2, 1, 5, 2, 1, 9, 1, 9, 6,
7, 4, 9, 8, 2, 3, 6, 5, 1, 3, 4, 5, 2, 4, 4, 1, 4, 2, 8, 6,
3, 1, 8, 6, 2, 2, 7, 9, 5, 1, 6, 9, 7, 3, 3, 1, 7, 8, 9, 8,
8, 9, 4, 9, 6, 2, 2, 6, 7, 8, 4, 6, 7, 8, 5, 9, 8, 9, 8, 7,
9, 6, 8, 6, 5, 9, 5, 9, 1, 8, 3, 6, 3, 8, 5, 6, 8, 6, 1, 9,
9, 3, 5]

unique = set(values)
```

# set

**Множество (set) в Python** — это изменяемая структура данных, которая содержит уникальные и неупорядоченные элементы.

**Некоторые свойства множеств:**

**Уникальность**. Каждый элемент множества неповторим. Не получится создать дубликат.
**Неупорядоченность**. У элементов множества нет порядкового номера.
**Изменяемость**. Можно добавлять во множество или удалять из него элементы.

**Про то, как реализован:** https://habr.com/ru/articles/830026/

hashable

immutable

| | List | Tuple | Dict | Set |
|---|---|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | | `mapping = {}`<br>`mapping = dict()` | `bag = {}`<br>`bag = set()` |
| Инициализация | `elements = [200, 300]`<br>`elements =`<br>`"2+3".split("2+3")` | `pair = (200, 300)`<br>`pair = tuple([200, 300])` | `mapping = {`<br>`"Аркадьич": 200,`<br>`"Борисыч": 300`<br>`}` | `bag = set([1, 2, 1, 2, 3])`<br>`bag = {1, 2, 3}` |
| Обращение к элементу | `v = elements[1]`<br>`v = elements[1:3]` | `v = pair[0]` | `v = elements["Аркадьич"]`<br>`v = elements.get("Ким", -1)` | ✗ |
| Добавление | `elements.append(400)` | ✗ | `v["Владимирович"] = 400` | `bag.add(100500)` |
| Расширение | `elements += [500, 600]` | ✗ | `elements.update(other_dict)` | `union, intersection, …` |
| Изменение | `elements[7] = "bob"`<br>`elements[1:2] = [0, 0]` | ✗ | `mapping[7] = "bob"`<br>`mapping["bob"] = 7` | ✗ |
| Проверка наличия | `if 200 in elements:` | `if 200 in pair:` | `if 200 in mapping.values():`<br>`if "bob" in mapping.keys():` | `if 200 in bag:` |
| Comprehension | `e = [_a for _a in elements]` | ✗ | `m = {_k: _v for _k, _v in`<br>`mapping.items()}` | `s = {_a for _a in elements}` |


{"val1"}
A set?
{"val1":"val2"}
A dictionary?!
imgflip.com

# ПРАКТИКА

# Что делать?

- https://docs.google.com/presentation/d/1Y6zULhpOT7Skd8yN26C4_kVsmNhVb6rMbHx-HlQrUo4/edit?usp=sharing
- Памятка: https://clck.ru/3GEng7
- Вопросы: https://t.me/dmi3eva

**[Коротко: https://goo.su/yqSdwF]**

# итоги

hashable

immutable

| | List | Tuple | Dict | Set |
|---|---|---|---|---|
| Инициализация пустым | `elements = []`<br>`elements = list()` | | `mapping = {}`<br>`mapping = dict()` | `bag = {}`<br>`bag = set()` |
| Инициализация | `elements = [200, 300]`<br>`elements =`<br>`"2+3".split("2+3")` | `pair = (200, 300)`<br>`pair = tuple([200, 300])` | `mapping = {`<br>`"Аркадьич": 200,`<br>`"Борисыч": 300`<br>`}` | `bag = set([1, 2, 1, 2, 3])`<br>`bag = {1, 2, 3}` |
| Обращение к элементу | `v = elements[1]`<br>`v = elements[1:3]` | `v = pair[0]` | `v = elements["Аркадьич"]`<br>`v = elements.get("Ким", -1)` | ✗ |
| Добавление | `elements.append(400)` | ✗ | `v["Владимирович"] = 400` | `bag.add(100500)` |
| Расширение | `elements += [500, 600]` | ✗ | `elements.update(other_dict)` | `union, intersection, …` |
| Изменение | `elements[7] = "bob"`<br>`elements[1:2] = [0, 0]` | ✗ | `mapping[7] = "bob"`<br>`mapping["bob"] = 7` | ✗ |
| Проверка наличия | `if 200 in elements:` | `if 200 in pair:` | `if 200 in mapping.values():`<br>`if "bob" in mapping.keys():` | `if 200 in bag:` |
| Comprehension | `e = [_a for _a in elements]` | ✗ | `m = {_k: _v for _k, _v in`<br>`mapping.items()}` | `s = {_a for _a in elements}` |

# Что общего?

**Коллекция** = объект, хранящий набор значений одного или различных типов

- **Built-In** = встроенные в интерпретатор
- Есть **общие методы**: `len(), …`
- **Итерируемые**
- Есть конструкторы
- Поддерживают преобразования друг в друга

# Итоги

- Обратная связь: https://otus.ru/polls/122210/

# Еще примеры
# List comprehension

# Разберём на детали

```
1    nums = [11, 2, 33, 24, 105]
2    target = []
3    for n in nums:
4        if n % 2 == 0:
5            value = n**2
6            target.append(value)
7    print(target)

>>> [4, 576]
```

# Разберём на детали

```
1   nums = [11, 2, 33, 24, 105]
2   target = []
3   for n in nums:
4       if n % 2 == 0:
5           value = n**2
6           target.append(value)
7   print(target)
```

Целевой список

```
>>> [4, 576]
```

# Разберём на детали

```
1    nums = [11, 2, 33, 24, 105]
2    target = []
3    for n in nums:
4        if n % 2 == 0:
5            value = n**2
6            target.append(value)
7    print(target)
```

Определение цикла

```
>>> [4, 576]
```

# Разберём на детали

```
1    nums = [11, 2, 33, 24, 105]
2    target = []
3    for n in nums:
4        if n % 2 == 0:
5            value = n**2
6            target.append(value)
7    print(target)
```

Условие выполнения

```
>>> [4, 576]
```

# Разберём на детали

```
1    nums = [11, 2, 33, 24, 105]
2    target = []
3    for n in nums:
4        if n % 2 == 0:
5            value = n**2
6            target.append(value)
7    print(target)
```

Расчет элемента

```
>>> [4, 576]
```

# Разберём на детали

```
1   nums = [11, 2, 33, 24, 105]
2   target = []
3   for n in nums:
4       if n % 2 == 0:
5           value = n**2
6           target.append(value)
7   print(target)
```

Целевой список

Определение цикла

Условие выполнения

Расчет элемента

```
>>> [4, 576]
```

# Структура list comprehension

```
1     nums = [11, 2, 33, 24, 105]
2     target = [n**2 for n in nums if n % 2 == 0]
```

# Структура list comprehension

```
1    nums = [11, 2, 33, 24, 105]
2    target = [n**2 for n in nums if n % 2 == 0]
```

Целевой список

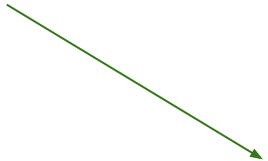
# Структура list comprehension

```
1    nums = [11, 2, 33, 24, 105]
2    target = [n**2 for n in nums if n % 2 == 0]
```

Расчет элемента

# Структура list comprehension

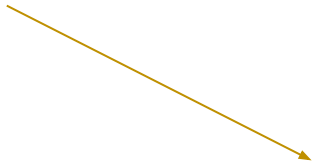
```
1    nums = [11, 2, 33, 24, 105]
2    target = [n**2 for n in nums if n % 2 == 0]
```



Определение цикла

# Структура list comprehension

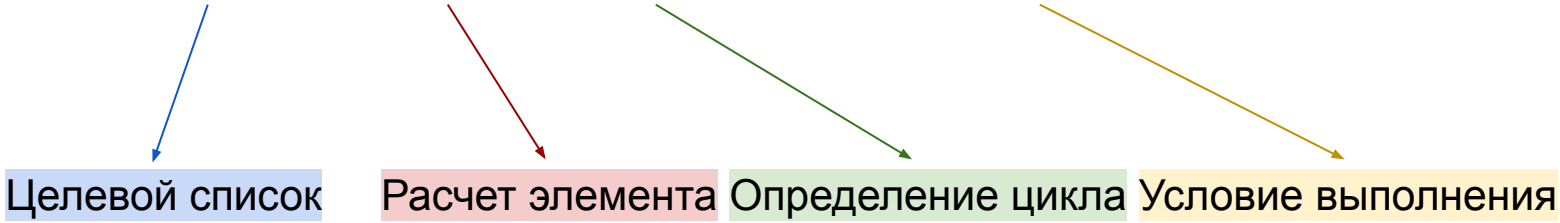
```
1    nums = [11, 2, 33, 24, 105]
2    target = [n**2 for n in nums if n % 2 == 0]
```



Условие выполнения

# Структура list comprehension

```
1    nums = [11, 2, 33, 24, 105]
2    target = [n**2 for n in nums if n % 2 == 0]
```



# Основной код сократился в 5 раз!

```
1    nums = [11, 2, 33, 24, 105]
2    target = []
3    for n in nums:
4        if n % 2 == 0:
5            value = n**2
6            target.append(value)
7    print(target)

>>> [4, 576]
```

```
1    nums = [11, 2, 33, 24, 105]
2    target = [n**2 for n in nums if n % 2 == 0]
3    print(target)

>>> [4, 576]
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Пример входных данных:**

```
words = ['style', 'uber', 'pet', 'electric', 'red']
```

**Пример выходных данных:**

```
result = [?]
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Пример входных данных:**

```
words = ['style', 'uber', 'pet', 'electric', 'red']
```

**Пример выходных данных:**

```
result = ['s', 'u', 'p', 'e', 'r']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 1:**

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = []
```

На Pascal:

```
for i:=1 to 5 do
 result[i] = words[i][0]
```

На C:

```
for (i = 0; i < 5; i++) {
 result[i] = words[i][0]
}
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 1:**

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = []

for word in words:
 result.append(_____)
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 1:**

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = []

for word in words:
 result.append(word[0])
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

```
result = []
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

```
result = ['s']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

```
result = ['s', 'u']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

```
result = ['s', 'u', 'p']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', ' electric', 'red']
result = [word[0] for word in words]
```

```
result = ['s', 'u', 'p', ' e']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', ' red']
result = [word[0] for word in words]
```

```
result = ['s', 'u', 'p', 'e', ' r']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

```
result = ['s', 'u', 'p', 'e', 'r']
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words]
```

**№1** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** этих слов.

**Способ 2** (в стиле Python!):

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [_w[0] for _w in words]
```

**№2** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых букв** слов, которые длиннее трёх букв.

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] for word in words if len(word) > 3]
```

**№3** Дан список слов английского алфавита (не длиннее 7 букв).
Требуется создать список **первых** букв слов, которые длиннее трёх букв, и **последних** букв остальных слов.

```
words = ['style', 'uber', 'pet', 'electric', 'red']
result = [word[0] if len(word) > 3 else word[-1] for word in words]
```