ПМ2 Прикладное программирование.

РО 2.1. Создавать консольные приложения на Python.

Тема 1. Условный оператор и логические выражения.

Цель занятия:

Познакомиться с логическими выражениями и условными операторами в Python.

Учебные вопросы:

1. Логические выражения.
2. Условный оператор.

1. Логические выражения.

Логическое выражение — это выражение, которое в результате вычисления принимает одно из двух значений:

- True — истинно
- False — ложно

Логические выражения используются в условных конструкциях (if, while) и для проверки условий в программах.

Операторы сравнения

Операторы сравнения позволяют сравнивать значения и возвращают True или False:

Оператор	Значение	Пример
==	Равно	2 == 4 (Результат: False)
!=	Не равно	2 != 4 (Результат: True)
>	Больше	2 > 4 (Результат: False)
<	Меньше	2 < 4 (Результат: True)
>=	Больше или равно	2 >= 4 (Результат: False)
<=	Меньше или равно	2 <= 4 (Результат: True)

Примеры:

```
x = 5
y = 10
print(x < 10)   # True
print(y > 10)   # False
print(y >= 10) # True
print(x == y)   # False
print(x != 10) # True
```

Логические операторы

Логические операторы позволяют объединять или изменять логические выражения, также возвращают True или False :

Оператор	Значение	Пример
and	Возвращает значение True если оба утверждения верны	x < 5 and x < 10
or	Возвращает True если одно из утверждений верно	x < 5 or x < 4
not	Меняет результат, возвращает False если результат True	not(x < 5 and x < 10)

Примеры:

```
print(True and True)    # True
print(True and False)   # False
print(True or False)    # True
print(False or False)   # False
print(not False)        # True
```

Приоритет выполнения

Когда несколько логических операторов встречаются вместе, Python выполняет их в следующем порядке:

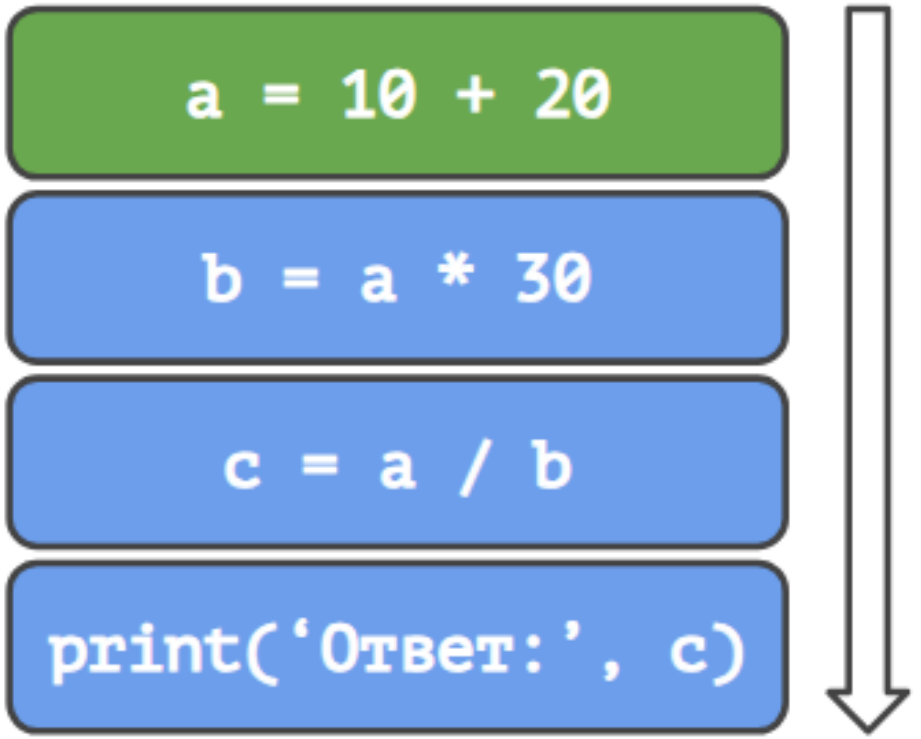
- операторы сравнения выполняются перед логическими операторами
- not — выполняется первым
- and — выполняется вторым
- or — выполняется последним

Примеры:

```
x = 5
y = 10
print(x > y or x == 10)      # True
print(not(x != 5))           # True
print(x > y and y > 0)       # False
```

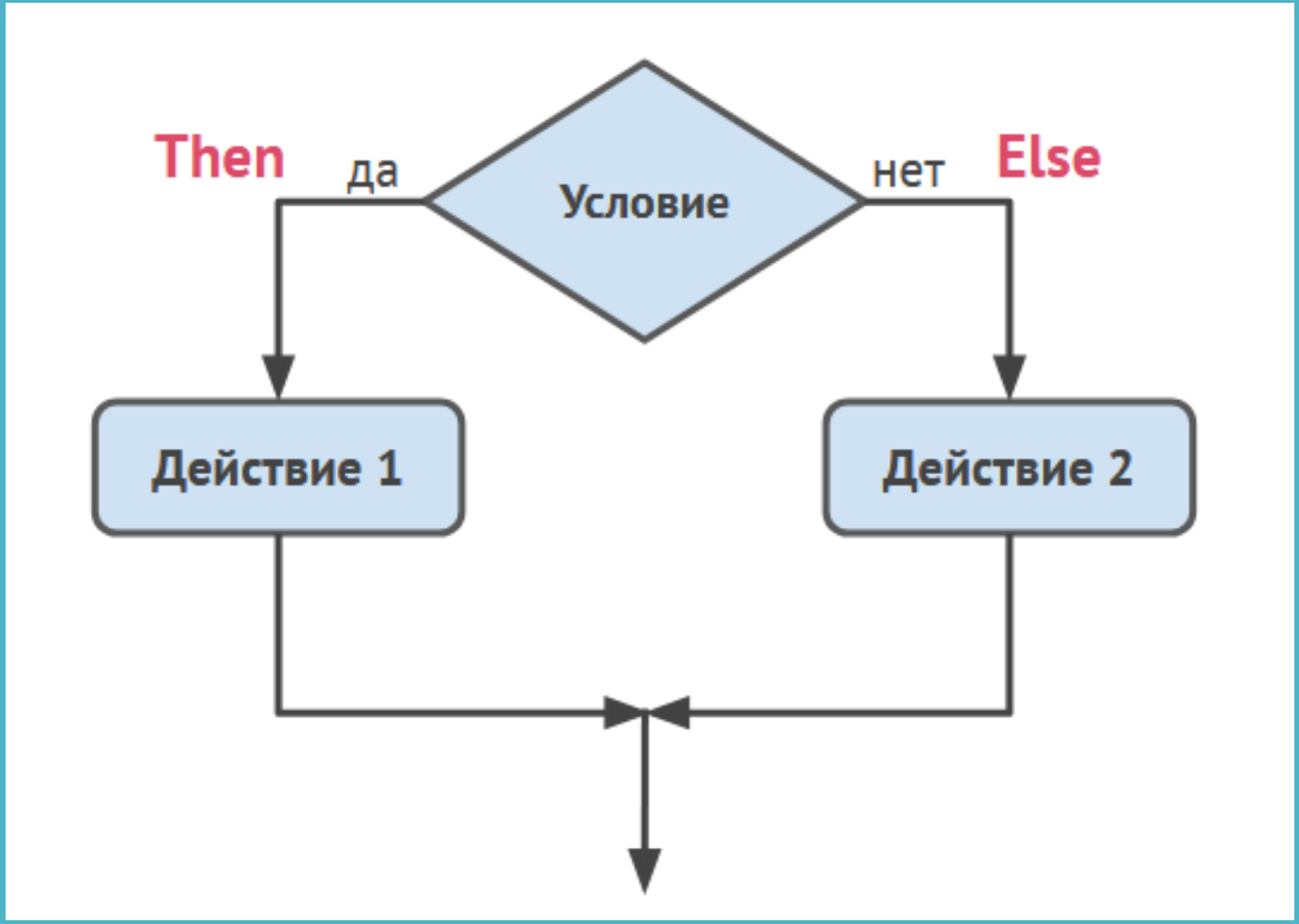
2. Условный оператор.

Как работает интерпретатор



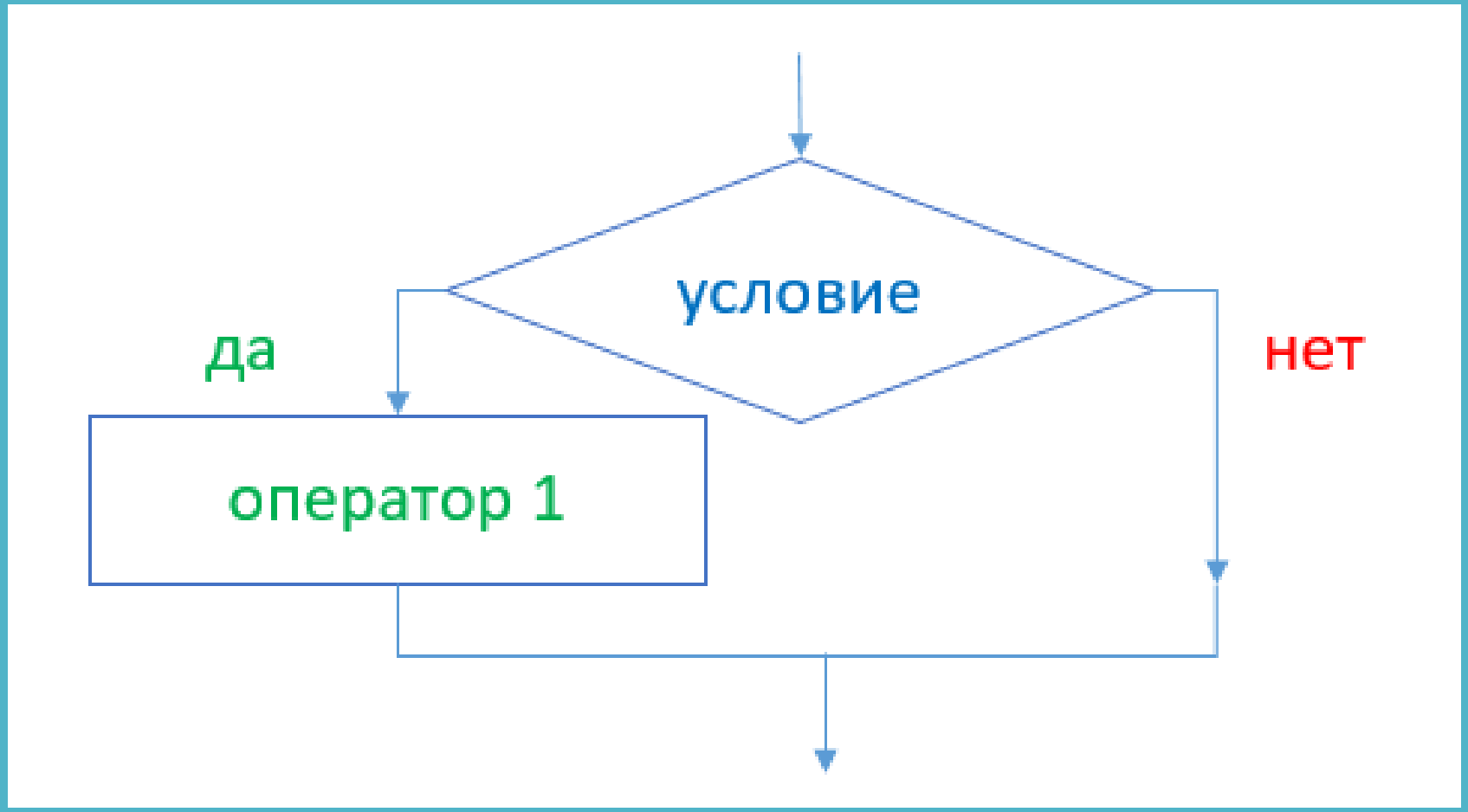
Он читает код и выполняет команды по очереди сверху вниз.

Условный оператор позволяет программе выполнять разные действия в зависимости от выполнения условия (истинности логического выражения).



Оператор **if**

Оператор if позволяет выполнять определённый блок кода только если выполняется условие.



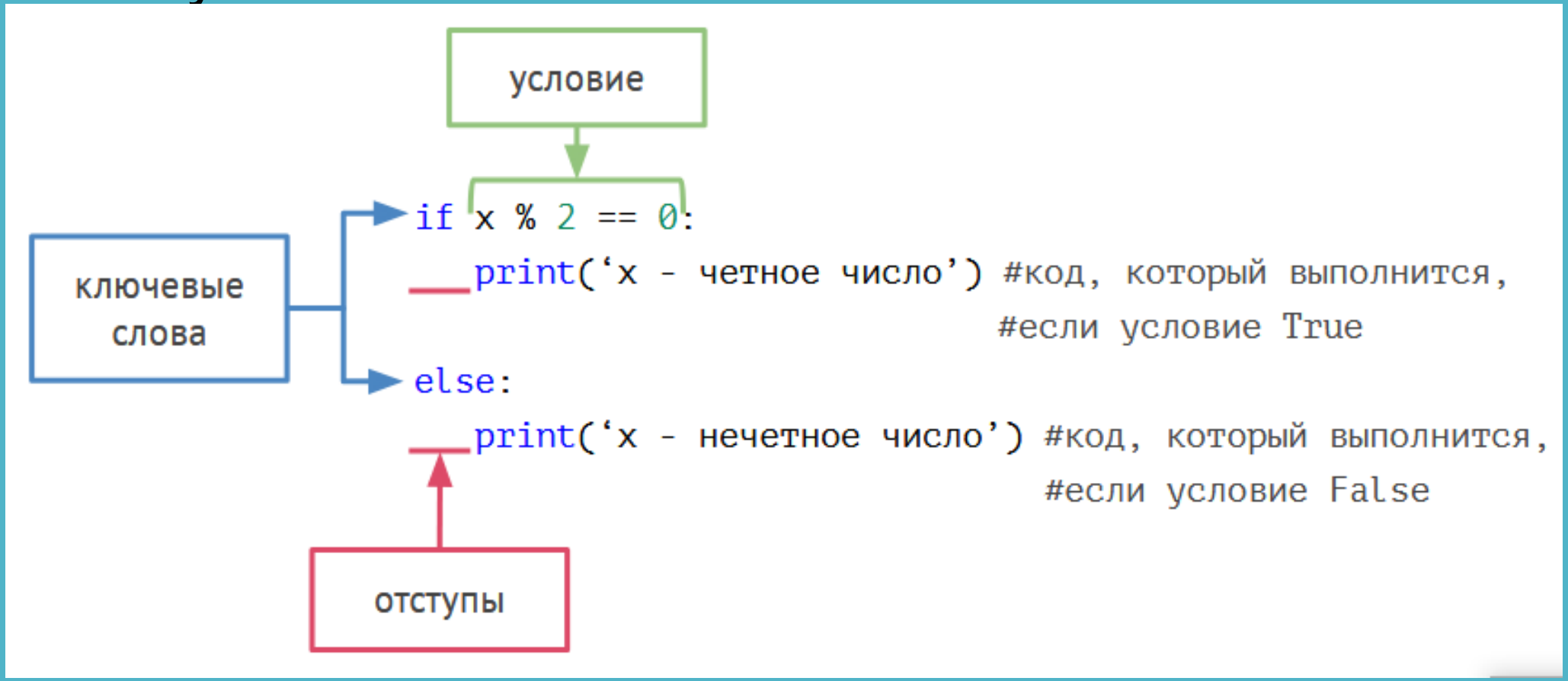
Пример:

```
x = 10
if x > 0:
    # код выполнится только если выражение True
    print("Число положительное")

print(x) # выполнится в любом случае
```

Оператор if…else

Если нужно, чтобы программа делала одно действие при истинном условии, а другое — при ложном, используют if…else



Пример:

```
x = 10
if x > 0:
    # код выполнится только если выражение True
    print("Число положительное")
else:
    # код выполнится только если выражение False
    print("Число отрицательное или 0")

print(x) # выполнится в любом случае
```

Оператор if … elif … else

Для проверки нескольких условий используется elif:

```
x = 10
if x > 0:
    print("Число положительное")
elif x == 0:
    print("Число равно 0")
else:
    print("Число отрицательное")
```

- Python проверяет условия сверху вниз
- Выполняется только первый блок с True
- Если ни одно условие не True → выполняется else

Вложенные условия (if внутри if)

Иногда нужно проверить дополнительное условие только если первое True:

```
is_male = True
age = 20
if is_male:
    if age >= 18:
        print("Призывник")
else:
    print("Не призывник")
```

Правила отступов в Python

Зачем нужны отступы?

- В Python отступы обозначают блоки кода.
- В отличие от других языков, здесь нет фигурных скобок {}.
- Программа понимает, какие команды относятся к условию, циклу или функции, по количеству пробелов в начале строки.

Общие правила:

- Вложенный блок кода выделяется отступом.
- Обычно используют 4 пробела на каждый уровень вложенности.
- Можно использовать табуляцию, но лучше только пробелы для единообразия.
- Блок кода начинается на новой строке после двоеточия :
- Пример: после if, elif, else, for, while, def и пр.

Тернарный оператор

Тернарный оператор — это условное выражение в одну строку, он выполняет ту же функцию, что и обычный **if…else**, но компактно.

Пример:

```
x = 10
result = "Положительное" if x > 0 else "Отрицательное"
print(result)  # Положительное
```

- Тернарный оператор короче и удобен для простых условий
- Для сложных условий лучше использовать обычный **if…else** с отступами

Выводы:

Логические выражения:

- Всегда возвращают True или False.
- Используются для проверки условий в программе.
- Основные операторы сравнения: ==, !=, >, <, >=, <=.
- Логические операторы: and, or, not.
- Приоритет выполнения: сначала сравнения, затем not, потом and, и в конце or.

Условный оператор if:

- Позволяет программе принимать решения и выполнять разные блоки кода в зависимости от условий.
- Основные формы:
- if — выполняется только при истинности условия
- if…else — добавляет альтернативу при ложном условии
- if…elif…else — позволяет проверять несколько условий последовательно
- Отступы — обязательны для определения блоков кода (обычно 4 пробела).
- Можно использовать вложенные условия, чтобы строить более сложные логики.

Контрольные вопросы:

- Что такое логическое выражение и какие значения оно может принимать?
- Какие существуют операторы сравнения и логические операторы в Python?
- Как работает условный оператор if и его расширения elif и else?
- Как использовать логические выражения внутри условий?
- Что такое тернарный оператор и как его применять?
- Как правильно организовывать вложенные условия и учитывать приоритет операторов?

Домашнее задание:

https://ru.hexlet.io/programs/python-basics-free

Материалы лекций:

https://github.com/ShViktor72/education2025

ПМ2_Прикладное_программирование

Обратная связь:

colledge20education23@gmail.com