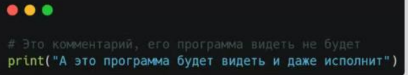
1. **Синтаксис и структура**

**Комментарии**

Строки, начинающиеся с символа # - это комментарии, примечания, которые автор программы пишет для других разработчиков

Питон игнорирует любые символы на строчке после #.



**Переменные и типы**

﻿﻿Переменная работает как подписанная коробка или помеченная ячейка, куда можно что-то положить и не потерять.

﻿﻿Когда в коде впервые пишут имя переменной, это называется объявление переменной. Имя придумывает сам программист.

Name = “Петя”

Age = 12

Print (name + ‘ ‘ + str(age) + ‘ годиков’ )



**Типы данных:**

Числовые типы данных:

﻿﻿int: Целые числа, например: × = 5

﻿﻿float: Числа с плавающей точкой, например: у = 3.14

Текстовые типы данных:

• str: Строки, состоящие из символов, заключенных в одинарные или двойные кавычки, например: пате = "Alice"

Логический тип данных:

• bool: Логические значения True (истина) или False (ложь).

например: is active = True

Списки и коллекции:

﻿﻿list: Упорядоченные изменяемые последовательности элементов, например: numbers = [1, 2, 3, 4]

﻿﻿tuple: Упорядоченные неизменяемые последовательности элементов,

например: coordinates = (x, y)

﻿﻿set: Неупорядоченные наборы уникальных элементов, например:

unique\_numbers = {1, 2, 3}

﻿﻿dict: Словари (ассоциативные массивы) для хранения пар "ключ-значение", например: person = ("name": "John", "age": 30}

None:

• NoneType: Особый тип данных, представляющий отсутствие значения, используется как значение по умолчанию или для указания отсутствия данных.

**Ввод данных**

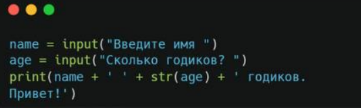
• Функция input() используется для получения данных от пользователя. Она приостанавливает выполнение программы, ожидая, пока пользователь введет текст, и после нажатия Enter возвращает этот текст в виде строки.

name = input( "Введите имя ")

age = input ("Сколько годиков? ")

print(name +' '+ str(age) +'годиков.

Привет!')



**Арифметические операторы**

Арифметические операторы:

+: Сложение

- Вычитание

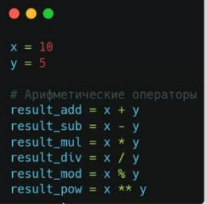
\*: Умножение

/: Деление (возвращает число с плавающей точкой)

//: Целочисленное деление (возвращает целое число)

%: Остаток от деления

\*\*: Возведение в степень



**Библиотека math**

Библиотека math в Python предоставляет разнообразные математические функции и константы для выполнения различных операций. Она включает в себя фУНКЦиИ ДлЯ работы с числами, вычисления тригонометрических функций, логарифмов, экспоненциальных функций и многое другое.



**Операторы сравнения**

Операторы сравнения:

==: Равно

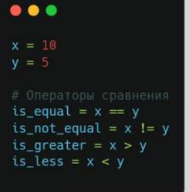
!=: Не равно

<; Меньше

>: Больше

<=: Меньше или равно

>=: Больше или равно



**Логические операторы и операторы присваивания**

Логические операторы:

﻿﻿and: Логическое И

﻿﻿ог: Логическое ИЛИ

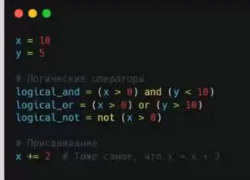
﻿﻿not: Логическое НЕ

Присваивание:

=: Присваивание значения

+=: Присваивание с добавлением

(x += 5 эквивалентно х = x + 5)



1. **Условные конструкции и циклы**

**Условные конструкции**

Условная конструкция в Python предоставляет возможность выполнения определенного блока кода, основываясь на результате проверки логического условия. В Python используется ключевые слова if, elif (сокращение от "else if") и else для создания условий.

if условие1:

# Код, выполняющийся, если условие1 истинно

elif условие2:

# Код, выполняющийся, если условие2 истинно

else:

# Код, выполняющийся, если ни одно из условий не истинно

If else:

x = 10

If x > 5:

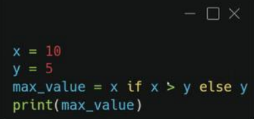
Print (“x больше 5”)

Else:

Print(“x не больше 5”)

**Тернарный оператор**

x if условие else у: Возвращает х, если условие истинно, и у в противном случае.

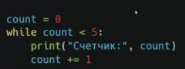


**Цикл while**

Цикл while в Python выполняет блок кода до тех пор, пока указанное условие остается истинным. Он проверяет условие перед каждой итерацией и завершает выполнение, как только условие становится ложным.

while условие:

# код, выполняемый до тех пор, пока условие истинно



**Цикл for**

Цикл for в Python используется для итерации (перебора) элементов в последовательности, такой как список, кортеж, строка или диапазон чисел. Он выполняет указанный блок кода для каждого элемента в последовательности.

for элемент і последовательность:

# код, выполняемый для каждого элемента



**break и continiue**

break и continue - это ключевые слова в Python, которые используются в циклах для управления выполнением цикла. Они позволяют вам контролировать, когда следует прервать выполнение всего цикла (break) или пропустить текущую итерацию и перейти к следующей

(continue).

1. break:

Когда оператор break встречается внутри цикла, выполнение цикла немедленно завершается, и программа продолжает выполнение после цикла. Это полезно, например, чтобы выйти из цикла при выполнении определенного условия.

for number in range(10):

if number == 5:

Break #выход из цикла, когда number становится равным 5

Print(number)

2. continue:

Оператор continue применяется внутри цикла и используется для пропуска текущей итерации цикла и перехода к следующей итерации.

Это может быть полезно, если вы хотите пропустить какие-либо операции в текущей итерации.

for number in range(10):

if number == 5:

continue # Пропустить операции при number равном перейти к следующей итерации

print (number )

**Функции**

Функции

Функции в Python определяются с использованием ключевого слова def, за которым следует имя функции и параметры в круглых скобках.

Вызов функции выполняется путем указания имени функции, за которым следуют

аргументы в круглых скобках.

def greet (name) :

print( "Привет, " + name + "!")

greet("Вася")

Возврат значений

Функции могут возвращать значения с помощью ключевого слова return.

def add(a, b):

result = a + b

return result

x = add(2,5)

print( "Результат: ", x)

Возврат нескольких значений

Python также позволяет возвращать несколько значений, которые будут упакованы в кортеж.

def calculate(a, b):

sum result = a + b

diff\_result = a - b

return sum\_result, diff\_result

sum\_value, diff\_value = calculate(10, 5)

print("Сумма:", sum\_value)

print( "Разность: ", diff\_value)

Возврат None

Если в функции нет оператора return или он указан без значения, функция вернет None.

def say\_hello(name) :

print("Привет, " + name + "!")

result = say\_hello ("Анна")

print(result) # Вывод: None

Аргументы функций

Функции могут принимать аргументы. В Python есть два типа аргументов: позиционные и именованные (ключевые). Позиционные аргументы передаются в том порядке, в котором объявлены параметры функции. Именованные аргументы передаются с указанием имени параметра.

def show\_info( name, age):

print( "Имя:", name )

print("Возраст:", age)

show\_info(age=25, name= "Андрей")

Аргументы по умолчанию

Функции могут иметь аргументы по умолчанию. Если аргумент не передан при вызове функции, будет использовано значение по умолчанию.

def greet (name="Гость"):

print( "Привет, " + name + "!")

greet( ) # Привет, Гость!

greet("Анна") # Привет, Анна!

Вложенные функции

В Python можно определять вложенные функции, которые находятся внутри других функций.

def outer\_function( ):

print( "Внешняя функция" )

def inner\_function( ):

print( "Внутренняя функция")

inner function( )

outer\_function( )

Локальные и глобальные переменные

Переменные, определенные внутри функции, называются локальными и видны только внутри этой функции. Переменные, определенные за пределами функции, называются глобальными и видны во всей программе.

# Глобальная переменная

global\_var = 10

def modify\_global( ):

global global\_var

global\_var += 1

def use\_ local( ):

local\_var = 5

print("Значение локальной переменной: " local var)

modify\_ global( )

print("Значение глобальной переменной после изменения:",global\_var)

use\_local()

# Попытка доступа к локальной переменной вне функции вызовет ошибку

# print("Значение локальной переменной вне функции: , "local\_var)

Рекурсия

Функция может вызывать саму себя. Это называется рекурсией.

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

print(factorial(4)) # Результат 24