

1. ОСНОВЫ ВЫЧИСЛЕНИЙ НА ЯЗЫКЕ PYTHON

1.1. Переменные и типы

Для хранения данных в программе используются переменные. Простыми словами, переменная – это некоторый объект, который имеет имя, тип и значение.

Правила для выбора имен переменных:

- имя должно состоять из букв, цифр и знаков подчеркивания, начинаться с буквы, не может совпадать с ключевыми словами языка;
- Python различает большие и маленькие буквы в имени (переменные **x** и **X** – разные переменные).

Также есть несколько рекомендаций, которые позволяют сделать программу более читабельной:

- имя переменной должно быть записано строчными буквами;
- имя переменной должно максимально точно соответствовать хранимым в ней данным;
- если имя переменной состоит из нескольких слов, то слова разделяются подчеркиванием.

Например, следующие имена переменных являются верными и могут использоваться для описания возраста человека:

```
age, user_age, person_age, user_1_age.
```

Имена следующих переменных записаны неверно:

12_age – начинается с цифры;

user-age – содержит недопустимый символ «-»;

f(x) – содержит недопустимые символы «(» и «)»;

if – является ключевым словом языка.

В Python не предусмотрено явное описание типа переменных, тип определяется в процессе выполнения программы в зависимости от данных, которые заносятся в переменную. В переменную можно занести:

- строки текста, которые в программе заключаются в двойные или одинарные кавычки ("Python", 'Я изучаю Python');
- целые значения (17, -4, 0);
- вещественные числа; при вводе значений этого типа в качестве разделителя целой и дробной части используется точка (-15.4, 12.0, 318.5302).

Задания для самопроверки

1. Выберите верные имена переменных:

```
name_use  
name+user  
1_user  
user_1_1  
user
```

2. Укажите тип, который соответствует следующим значениям (для строковых данных – **string**, для целых чисел – **integer**, для вещественных **float**, если значение записано неверно – укажите **error**):

```
"Привет"  
1,67  
-7.90  
0  
24  
12/09/2022
```

1.2. Основные конструкции языка

1.2.1. Операторы ввода и вывода

Для ввода данных используется оператор `input()`, который считывает строку текста, введенную пользователем с консоли, и заносит ее в переменную. Следующий оператор вводит строку текста и заносит ее в переменную `name`:

```
name = input()
```

Для вывода значения переменной используется оператор `print()`, оператор выведет на экран значение переменной `name`:

```
print(name)
```

С помощью оператора `print()` можно выводить строки в кавычках и значения переменных одновременно, при выводе они разделяются запятой.

```
print("Привет, ", name, "!")
```

Для ввода числовых значений в переменные необходимо перед оператором ввода `input()` указать тип значения (`int` или `float`). Следующий оператор вводит целое значение в переменную `age`:

```
age = int(input())
```

Кроме того, Python допускает использование подсказки пользователю непосредственно в операторе `input()`. При использовании такой подсказки ввод данных пользователем при выполнении программы будет осуществляться сразу после текста подсказки на той же строке. Следующий оператор вводит целое значение в переменную `age`, выдавая пользователю подсказку:

```
age = int(input("Сколько Вам полных лет? "))
```

Пример. Спросить у пользователя его имя и сколько ему лет. Вывести приветствие пользователю.

Код:

```
name = input("Ваше имя? ")
age = int(input("Сколько Вам полных лет? "))
print("Привет, ", name, "! Ваш возраст", age, ".")
```

Результат:

```
Ваше имя? Алина
Сколько Вам полных лет? 20
Привет, Алина ! Ваш возраст 20 .
```

Задания для самопроверки

1. Все перечисленные ниже операторы являются неверными, укажите ошибку и исправьте каждый оператор:

```
name = print("Ваше имя? ")
name = input("Ваше имя? ")
x = input(int("x="))
print("x=" x)
```

2. Напишите программу, которая спрашивает у пользователя его имя, возраст и рост (в метрах). Вывести приветствие пользователю, указав его возраст и рост.

1.2.2. Оператор присваивания

Вычисление выражения и занесение его значения в некоторую переменную осуществляется с помощью оператора присваивания. Например, следующий оператор заносит в переменную `name` значение «Алина»:

```
name = "Алина"
```

В выражениях допустимо использовать арифметические операции:

- сложение «+»,
- вычитание «-»,
- умножение «*»,
- деление «/»,
- целочисленное деление «//»,
- остаток от деления «%», возведение в степень «**».

Python поддерживает привычный порядок выполнения операций: сначала возведение в степень, затем умножение, деление, целочисленное деление и остаток (все они имеют одинаковый приоритет, применяются последовательно слева направо), последним выполняется сложение или вычитание. Для изменения приоритета используются круглые скобки.

Примеры выражений:

```
3 + 2 * 5 ** 2 - 1 = 52
(3 + 2) * 5 ** 2 - 1 = 124
5 ** 2 / 2 = 12.5
63 % 10 = 3
63 % 10 + 4 = 7
63 % (10 + 4) = 7
63 // 10 = 6
63 // 10 + 4 = 10
63 // (10 + 4) = 4
```

Пример. Спросить у пользователя его имя и его год рождения. Вычислить сколько лет сейчас пользователю и вычислить результат (считаем, что текущим является 2022 год).

Код:

```
name = input("Ваше имя? ")
year_birth = int(input("В каком году Вы родились? "))
age = 2022 - year_birth
print("Привет, ", name, "! Ваш возраст ", age, ".")
```

Результат:

```
Ваше имя? Алина
В каком году Вы родились? 2002
Привет, Алина ! Ваш возраст 20 .
```

Задания для самопроверки

1. Вычислите значения следующих выражений для заданных x и y :

```
x = 5
y = 3
x + y ** 2 =
y ** 2 * x =
x ** y - x =
x ** (y - 2) =
x / y + 3 =
x / y * 3 =
x // y =
x % y =
```

2. Запишите операторы присваивания, которые описывают следующие выражения:

$$y = 4 \cdot x^2 - 5$$

$$z = \frac{(x - 2)^2 - 1}{x + 4}$$

1.2.3. Условный оператор

Для реализации возможности выполнения различных действий в зависимости от условия в Python используется условный оператор **if...elif...else**, синтаксис которого следующий (разделы **elif** и **else** могут отсутствовать):

```
if условие_1:
    операторы_1
elif условие_2:
    операторы_2
...
else:
    операторы
```

Выполняется этот оператор так:

- проверяется условие_1 после **if**, если условие истинно, то выполняются операторы_1;
- в противном случае проверяется условие_2 после первого **elif**, если оно истинно, то выполняются операторы_2;
- аналогично обрабатываются все **elif**;
- в случае если все условия ложны, то выполняются операторы после **else**.

Особенности синтаксиса оператора if:

1. ключевые слова (**if**, **elif**, **else**) записываются строчными буквами;
2. после условия в операторах **if** и **elif**, а также после **else** обязательно ставится знак двоеточия «:», который означает наличие вложенных конструкций;
3. все операторы после знака «:» считаются вложенными и записываются с отступом (ставится два или четыре пробела перед каждым вложенным оператором).

Вложенные инструкции в Python записываются в соответствии с одним и тем же шаблоном, когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, с отступом под строкой основной конструкции.

Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков (обычно это 4 пробела).

В следующем примере оператор 1 и оператор 2 относятся к вложенному блоку основной конструкции, а оператор 3 находится на уровне основной конструкции.

```
основная_конструкция:
    оператор_1
    оператор_2
оператор_3
```

Условия в операторе **if** являются логическими выражениями и могут включать:

- знаки отношений (равно «==» (два подряд равенства), не равно «!=», больше «>», меньше «<», больше или равно «>=», меньше или равно «<=»);
- логические операции (И «and», ИЛИ «or», НЕ «not»);

Результатом вычисления логических выражений является либо Истина (True), другими словами, в этом случае выражение «верно», либо Ложь (False) – выражение «неверно».

При использовании логической операции И (and) выражение верно, если все его элементы верны. При использовании логической операции ИЛИ (or) выражение верно, если хотя бы один его элемент верен.

Примеры:

```
3 > 5 результат False
3 == (1 + 2) результат True
3 >= (1 + 2) результат True
3 != (1 + 2) результат False
```

```
3 < 5 and 3 >= 4 результат False
3 < 5 or 3 >= 4 результат True
3 < 5 and not (3 >= 4) результат True
```

Пример. Дана сторона квадрата **a** (вещественное число). Если введено положительное значение **a**, то посчитать площадь и периметр квадрата. В противном случае вывести сообщение об ошибке.

Код:

```
a = float(input("a = "))
if a <= 0:
    print("Длина должна быть положительной")
else:
    p = 4 * a
    s = a * a
    print("p =", p, "s =", s)
```

Результат:

Тест 1

a = 4

p = 16.0 s = 16.0

Тест 2

a = -2

Длина должна быть положительной

1.2.4. Комментарии в Python

Комментарии – это способ описания того, что делает программа на самом высоком уровне. Это специальные отмеченные строчки, которые комментируют код. Это неисполняемые, но все равно важные части кода. Комментарии делают код более понятным и читаемым.

В Python строка с комментарием начинается с символа «#». Все, что написано после этого символа, при выполнении программа пропускает.

Например, если в программе встречается следующий оператор:

```
income = x * (1 + k / (12 * 100)) ** n - x
```

совершенно непонятно, что именно он делает. Если же добавить комментарий, то его назначение становится прозрачным:

```
#доход от вклада x на n месяцев под k% годовых с капитализацией процентов
income = x * (1 + k / (12 * 100)) ** n - x
```

1.2.5. Встроенные функции Python

В языке Python реализовано много встроенных функций, использующихся для различных целей, вот некоторые из них:

`abs(x)` – вычисляет модуль числа `x`;

`round(x, n)` – округляет число `x` до `n` знаков после запятой;

`min(x, y, z, ...)` – находит минимальное из списка значений;

`max(x, y, z, ...)` – находит максимальное из списка значений.

Примеры:

```
abs(-4) = 4
```

```
abs(5) = 5
```

```
round(3.141569) = 3
```

```
round(3.141569, 4) = 3.1416
```

```
min(3, 4, 6.8, -3, -1.89) = -3
```

```
max(3, 4, 6.8, -3, -1.89) = 6.8
```

Важно помнить, что в языке Python для округления чисел используется банковский способ округления, который отличается от математического, если в следующей позиции после позиции округления стоит 5. При математическом округлении в этом случае число

округляется в большую по модулю сторону, при банковском – к ближайшему чётному. Например:

```
round(2.65, 1) = 2.6
round(2.75, 1) = 2.8
```

Пример. Всего в группе учатся **n** студентов, из них проживает в общежитии **k** человек. Проверить входные данные и вывести процент студентов, проживающих в общежитии. При выводе результат округлить до двух знаков после запятой.

Код:

```
n = int(input("n = "))
k = int(input("k = "))
# количества студентов - положительное число, также количество
# проживающих в общежитии не может быть больше общего количества
if n <= 0 or k <= 0 or k > n:
    print("Ошибочные входные данные")
else:
    percent = k / n * 100
    print("проживает в общежитии:", round(percent, 2), "%")
```

Результат:

```
тест 1
n = 23
k = 17
проживает в общежитии: 73.91 %
тест 2
n = 20
k = 21
Ошибочные входные данные
```

Задания для самопроверки

1. Какое значение получит переменная **y** после программы:

```
x = 3
if x <= 0:
    y = x
elif x <= 5:
    y = x ** 2
else:
    y = -x
```

2. Вычислите значения следующих логических выражений:

```
x == y or (x - 4) >= 1
x != 2 and y <= 5
```

3. Какие значения получатся после выполнения следующих встроенных функций?

```
round(3.89) =
round(-2.49) =
round(4.999, 2) =
abs(-5) =
abs(6.1) =
max(1.9, -2, -3.1, 0.2) =
abs(min(3, -2, -3.1, 0.2)) =
min(3, abs(-2), abs(-3.1), 0.2) =
```

4. Напишите программу для решения следующей задачи:

Ракета запускается с точки на экваторе и развивает скорость v км/с. Каков результат запуска?

Указание: если $v \leq 7.8$ км/с, то ракета упадет на Землю (вывести 0), если $7.8 < v < 11.2$, то ракета станет спутником Земли (вывести 1), если $11.2 \leq v \leq 16.4$, то ракета станет спутником Солнца (вывести 2), если $v > 16.4$, то ракета покинет Солнечную Систему (вывести 3).

Если будет введено значение, меньшее или равное 0, то вывести "error".