IT CAREER HUB

Урок 2.
Базовые типы данных, работа с функциями

IT CAREER HUB

Тип данных и объект

Тип данных определяет, какие значения может хранить объект и какие операции можно выполнять с этими значениями.

Объект в Python — это любое значение, с которым можно работать в программе.

Всё в Python — это объект: числа, строки, списки и т.д. Каждый объект хранит данные и имеет определённый тип, который определяет, что это за данные и какие операции можно с ними выполнять.

IT CAREER HUB

Динамическая типизация

Python является динамически типизированным языком, что означает, что тип переменной определяется автоматически в момент присваивания значения, и он может меняться в процессе выполнения программы.

Тип данных присваивается не переменной, а объекту, на который она ссылается. Это позволяет одной и той же переменной в ходе программы ссылаться на объекты разных типов данных.

Пример

```python
price = 10                # Тип автоматически определен как int
has_passed = 9.99         # Тип автоматически переопределен как float
```

IT CAREER HUB

Примитивные типы данных

Примитивные типы данных в Python — это основные типы, которые используются для хранения простых (единичных) значений. В Python примитивные типы данных включают: целые числа, числа с плавающей запятой, строки, логические значения, отсутствие значения.

Целые числа (int) представляют собой значения без дробной части. Они могут быть положительными, отрицательными или равны нулю.

Пример

```python
x = 5         # Положительное целое число
y = -3        # Отрицательное целое число
z = 0         # Ноль
```

Числа с плавающей запятой (float) представляют собой значения с дробной частью.

Пример

```python
pi = 3.14              # Число с плавающей запятой

temperature = -5.2     # Отрицательное число с плавающей запятой
```

Строки (str) представляют собой упорядоченные последовательности символов, заключённых в одинарные или двойные кавычки, а также в тройные кавычки (как одинарные, так и двойные). Тройные кавычки позволяют создавать многострочные строки, сохраняющие исходное форматирование, включая переносы строк.

Пример

```python
# Одинарные кавычки
single_quote_string = 'Это строка в одинарных кавычках.'

# Двойные кавычки
double_quote_string = "Это строка в двойных кавычках."

# Одинарные тройные кавычки (многострочная строка)
triple_quote_string = '''Это строка в
тройных кавычках, которая
может занимать несколько строк.'''

# Альтернативный вариант с двойными тройными кавычками
another_triple_quote_string = """Это ещё одна
многострочная строка, но с использованием
двойных тройных кавычек."""
```

Логические значения (bool) представляют два возможных состояния: `True` (истина) и `False` (ложь). Они записываются текстом с большой буквы без кавычек.

Пример

```python
is_student = True          # Логическое значение True
has_passed = False         # Логическое значение False
```

Тип NoneType представляет отсутствие значения. Единственное значение для этого типа: None.

Пример

```python
value = None               # Переменная, не содержит никакого значения
```

Значение None часто используется для инициализации переменных, когда значение ещё не известно, или для обозначения того, что функция ничего не возвращает.

Эти примитивные типы данных позволяют эффективно хранить и обрабатывать информацию в программах на Python. Они не могут быть разбиты на более мелкие части.

IT CAREER HUB

Задание для закрепления 1

1. Какой тип данных будет у переменной x, если присвоить ей значение `x = 5.0`?

a. `str`
b. `bool`
c. `int`
d. `float`

Посмотреть ответ

2. Что из перечисленного является логическим значением?

a. `"True"`
b. `0`
c. `True`
d. `None`

Посмотреть ответ

3. Что произойдёт, если переменной сначала присвоить целое число, а затем строку?

a. Возникнет ошибка
b. Тип переменной изменится автоматически
c. Программа завершится
d. Значение переменной будет преобразовано в `int`

Посмотреть ответ

IT CAREER HUB

Возвращаемое значение

Возвращаемое значение — это результат, который функция передаёт обратно в место своего вызова. Функции могут возвращать значение, или ничего не возвращать, в последнем случае по умолчанию будет возвращено None.

IT CAREER HUB

Процедуры в Python

В Python нет отдельного понятия "процедура", как в некоторых других языках программирования. Однако термин **процедура** часто используется для описания функций, которые **ничего не возвращают** (или возвращают None) и выполняют определённые действия, такие как вывод на экран или модификация данных.

Пример процедуры

```python
print("Привет, мир!")  # Выводим сообщение, но print не возвращает результат

Если попытаться вывести результат работы процедуры, результат всегда будет None.

value = print("Привет, мир!")  # Выводит сообщение "Привет, мир!", но print не возвращает результат

print(value)                   # Выводит None
```

IT CAREER HUB

Передача аргументов в функцию

В функцию можно передавать аргументы разными способами:

1. Значения (литералы), например, числа или строки:

```python
print(5)
print("Привет, мир!")
```

2. Переменные, из которых будут получены значения:

```python
text = "Привет, мир!"
print(text)
```

3. Результат выполнения математических операций:

```python
print(2 + 5)
```

4. Результаты выполнения других функций:

```python
print(input("Введите имя: "))
```

Т.к. `input` возвращает то, что ввел пользователь, то это значение можно передать в функцию `print`.

IT CAREER HUB

Функция type

Функция `type` используется для определения типа объекта. Она возвращает тип данных (класс), который находится в переменной в момент выполнения функции `type`. Чтобы увидеть результат вызова функции `type`, необходимо передать его как аргумент в функцию `print`.

Пример

Python

```python
x = 10

print(type(x))  # <class 'int'>

x = "Привет"

print(type(x))  # <class 'str'>
```

IT CAREER HUB

Задание для закрепления 2

Что произойдёт, если передать переменную в функцию `print()`?

a. Будет выведено имя переменной
b. Будет выведено значение переменной
c. Возникнет ошибка
d. Будет выведен тип переменной

Посмотреть ответ

Конкатенация

Конкатенация — это процесс объединения двух или более строк в одну строку.

Для этого используется оператор +. Это позволяет создавать новые строки путём добавления одной строки к другой. Это может быть полезно для комбинирования данных или формирования строки для вывода из нескольких элементов.

Пример

```python
str1 = "Привет"

str2 = "мир"

result = str1 + " " + str2  # Объединение строк с пробелом между ними

print(result)  # Вывод: Привет мир
```

Умножение строк позволяет создать новую строку, состоящую из повторений исходной строки. Это делается с использованием оператора *, за которым следует число, указывающее, сколько раз строка должна быть повторена. Умножение может быть полезно для создания разделителей, повторяющихся шаблонов текста и других целей.

Пример

```python
str1 = "Hi! "
result = str1 * 3  # Создание строки из трёх повторений исходной строки
print(result)  # Вывод: Hi! Hi! Hi!
```

Обратите внимание, что при использовании этих операций изначальные строки не изменяются.

IT CAREER HUB

Работа с кавычками в строках

В Python строки можно заключать в одинарные (') или двойные кавычки ("), что позволяет гибко работать с текстом.

Эти два варианта используются взаимозаменяемо, но есть особенности, когда лучше использовать определенные кавычки.

Когда использовать одинарные и двойные кавычки? Если внутри строки необходимо использовать кавычки, то целесообразно выбирать противоположные по типу кавычки для обрамления строки.

Примеры

```python
# Одинарные кавычки внутри строки
quote1 = "Она сказала: 'Привет!'"

# Двойные кавычки внутри строки
quote2 = 'Он ответил: "Привет!"'
```

В таких случаях это избавляет от необходимости экранировать кавычки.

Для многострочных строк и строк, содержащих как одинарные, так и двойные кавычки, удобно **использовать тройные кавычки** (''' или """). В данном случае внутри строки можно использовать как одинарные, так и двойные кавычки без необходимости экранирования.

Пример многострочной строки

```python
multi_line_string = """Это многострочная строка, в которой есть 'одинарные' и "двойные" кавычки."""
```

IT CAREER HUB

Экранирование символов и кавычек

Экранирование — это способ обработки специальных символов внутри строк.

В Python используется обратный слэш (\) для обозначения специальных символов и создания **escape-последовательностей**. Эти последовательности представляют собой комбинации символов для корректного отображения таких символов, как кавычки, обратный слэш, перенос строки или табуляцию, позволяя включать их в строку без конфликтов с синтаксическими правилами.

Если строка обрамлена в одинарные кавычки, а внутри нужно использовать одинарную кавычку, её нужно экранировать. То же правило действует и для двойных кавычек.

Пример экранирования кавычек

```python
# Одинарные кавычки внутри строки

string1 = 'В строке есть \'одинарные\' и "двойные" кавычки.'

print(string1)



# Двойные кавычки внутри строки

string2 = "В строке есть 'одинарные' и \"двойные\" кавычки."

print(string2)
```

Экранирование также используется для вставки специальных символов, таких как:

- \n — новая строка
- \t — табуляция

Пример:

```python
# Новая строка

text = "Первая строка\nВторая строка"

print(text)

# Табуляция

text_with_tab = "Первая строка\tВторая строка"

print(text_with_tab)
```

Если в строке необходимо вставить обратный слэш, его тоже нужно экранировать, так как сам обратный слэш является специальным символом.

Пример

```python

path = "C:\\Users\\Username\\Documents"
```

IT CAREER HUB

Параметры sep и end

Функция **print** имеет несколько необязательных параметров, которые позволяют настроить ее поведение. Вот основные из них:

1. **sep** (разделитель): Определяет строку, которая будет вставлена между аргументами. По умолчанию используется пробел.

Пример

```python
print("one", "two", "three")                # one two three
print("one", "two", "three", sep="---")     # one---two---three
```

2. **end** (конец): Определяет строку, которая будет добавлена в конец вывода. По умолчанию используется символ новой строки `\n`.

Пример

```python
print("Привет,", "мир", end="!!")
print("Привет")
```

IT CAREER HUB

Задание для закрепления 3

1. Какой результат выведет следующий код?

```python
str1 = "Привет"
str2 = "мир"
result = str1 + str2
print(result)
```

a. `Привет мир`
b. `Приветмир`
c. `Привет, мир`
d. `Ошибка`

Посмотреть ответ

2. Какой результат выведет следующий код?

```python
str1 = "Hi! "
result = str1 * 3
print(result)
```

a. `Hi! Hi! Hi!`
b. `HiHiHi`
c. `Ошибка`
d. `Hi!`

Посмотреть ответ

IT CAREER HUB

Ответы на задания

Задания на закрепление 1	Вернуться к заданиям
1. Тип данных	Ответ: d
2. Логическое значение	Ответ: c
3. Порядок присвоения	Ответ: b
Задания на закрепление 2	Вернуться к заданиям
Передача переменной в функцию print()	Ответ: b
Задания на закрепление 3	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
2. Результат выполнения кода	Ответ: a

IT CAREER HUB

Практическая работа

1. Приветствие с восклицанием

Напишите программу, которая выведет строку "Привет, мир!" с использованием параметра end, чтобы вывод закончился на восклицательном знаке вместо новой строки.

Пример вывода:

```
Привет, мир!
```

2. Цитата

Создайте программу, которая выведет строку: She said: "It's amazing!" двумя способами (одинаковый результат, но разный код).

Пример вывода:

```
She said: "It's amazing!"
```

```
She said: "It's amazing!"
```