СОЗДАНИЕ КОДА С ПОМОЩЬЮ PYTHON

VS Code - это особый тип текстового редактора, который называется компилятором. Вверху вы увидите текстовый редактор, а внизу вы увидите терминал, где вы можете выполнять команды.

В терминале вы можете выполнить code hello.py, чтобы начать кодирование.

В текстовом редакторе, приведенном выше, вы можете ввести print("hello, world"). Это известная каноническая программа, которую почти все программисты пишут в процессе обучения.

В окне терминала вы можете выполнять команды. Чтобы запустить эту программу, вам нужно переместить курсор в нижнюю часть экрана, щелкнув в окне терминала. Теперь вы можете ввести вторую команду в окне терминала. Рядом со знаком доллара введите python hello.pyи нажмите клавишу enter на клавиатуре.

Напомним, компьютеры действительно понимают только нули и единицы. Поэтому при запуске python hello.pypython интерпретирует текст, который вы создали, hello.py и преобразует его в нули и единицы, которые может понять компьютер.

Результатом выполнения python hello.py программы является hello, world.

Поздравляю! Вы только что создали свою первую программу.

ФУНКЦИИ

Функции - это глаголы или действия, которые компьютер или компьютерный язык уже знают, как выполнять.

В вашей hello.py программе print функция знает, как печатать в окне терминала.

Print Функция принимает аргументы. В данном случае "hello, world" это аргументы, которые print принимает функция.

ОШИБКИ

Ошибки являются естественной частью программирования. Это ошибки, проблемы, которые вам предстоит решить! Не отчаивайтесь! Это часть процесса становления великим программистом.

Представьте, что в нашей hello.py программе случайно набрано print("hello, world" уведомление о том, что мы пропустили финал), требуемый компилятором. Если я намеренно допущу эту ошибку, то компилятор выдаст ошибку в окне терминала!

Часто сообщения об ошибках информируют вас о вашей ошибке и дают вам подсказки о том, как их исправить. Однако будет много случаев, когда компилятор не такого типа.

Улучшение вашей первой программы на Python

Мы можем персонализировать вашу первую программу на Python.

В нашем текстовом редакторе hello.py мы можем добавить еще одну функцию. input это функция, которая принимает приглашение в качестве аргумента. Мы можем отредактировать наш код, чтобы сказать

```
input("What's your name? ")

print("hello, world")
```

Однако это редактирование само по себе не позволит вашей программе выводить то, что вводит ваш пользователь. Для этого нам нужно будет познакомить вас с переменными

ПЕРЕМЕННЫЕ

Переменная - это просто контейнер для значения в вашей собственной программе.

В вашей программе вы можете ввести свою собственную переменную в свою программу, отредактировав ее следующим образом

```
name = input("What's your name? ")

print("hello, world")
```

Обратите внимание, что этот =знак равенства в середине name = input("What's your name? ")имеет особую роль в программировании. Этот знак равенства буквально присваивает то, что находится справа, тому, что находится слева. Следовательно, значение, возвращаемое by input("What's your name? "), присваивается name.

Если вы отредактируете свой код следующим образом, вы заметите ошибку

name = input("What's your name? ")

print("hello, name")

Программа вернется hello, nameв окне терминала независимо от того, что вводит пользователь.

При дальнейшем редактировании нашего кода вы можете ввести

name = input("What's your name? ")

```
print("hello,")

print(name)
```

Результат в окне терминала будет

```
What's your name? David

hello

David
```

Вы можете узнать больше в документации Python о типах данных.

КОММЕНТАРИИ

Комментарии - это способ для программистов отслеживать, что они делают в своих программах, и даже информировать других о своих намерениях в отношении блока кода. Короче говоря, это заметки для себя и других, которые увидят ваш код!

Вы можете добавлять комментарии к своей программе, чтобы иметь возможность видеть, что делает ваша программа. Вы можете отредактировать свой код следующим образом:

```
# Ask the user for their name

name = input("What's your name? ")

print("hello,")

print(name)
```

Комментарии также могут служить для вас списком дел.

ПСЕВДОКОД

Псевдокод - это важный тип комментариев, который становится особым типом списка дел, особенно когда вы не понимаете, как выполнить задачу кодирования. Например, в вашем коде вы можете отредактировать свой код, чтобы сказать:

```
# Ask the user for their name

name = input("What's your name? ")


# Print hello

print("hello,")
```

```
# Print the name inputted
print(name)
```

Дальнейшее совершенствование вашей первой программы на Python

Мы можем дополнительно отредактировать наш код следующим образом:

```
# Ask the user for their name
name = input("What's your name? ")

# Print hello and the inputted name
print("hello, " + name)
```

Оказывается, некоторые функции принимают много аргументов.

Мы можем использовать запятую, для передачи нескольких аргументов, отредактировав наш код следующим образом:

```
# Ask the user for their name
name = input("What's your name? ")

# Print hello and the inputted name
print("hello,", name)
```

Вывод в терминале, если бы мы набрали “David”, мы были бы hello, David. Успех.

Строки и пареметры

Строка, известная в Python как str, представляет собой последовательность текста.

Если немного перемотать наш код назад, то можно заметить, что результат выводится на нескольких строках:

```
# Запросить у пользователя его имя
name = input("Как вас зовут? ")
print("hello,")
```

print(name)

Функции принимают аргументы, которые влияют на их поведение. Если мы посмотрим на документацию к функции print, то заметим, что можем узнать много нового об аргументах, которые принимает функция print.

Изучив документацию, вы узнаете, что функция print автоматически включает в себя часть кода end='\n'. Это \n указывает на то, что функция print будет автоматически создавать разрыв строки при запуске. Функция print принимает аргумент end` и по умолчанию создает новую строку.

Однако, технически мы можем сами задать аргумент end так, чтобы новая строка не создавалась!

Мы можем изменить наш код следующим образом:

Запросить у пользователя его имя

name = input("Как вас зовут? ")

print("hello,", end="")

print(name)

Указав end ="", мы переписываем значение по умолчанию end, чтобы никогда не создавать новую строку после первого оператора print. Если задать имя как "David", то в окне терминала появится сообщение hello, David.

Параметры - это аргументы, которые могут быть приняты функцией.

Вы можете узнать больше в документации Python по print.

Небольшая проблема с кавычками

Обратите внимание, как сложно добавить кавычки в строку.

print("hello, "friend") не сработает, и компилятор выдаст ошибку.

В общем, есть два подхода к исправлению этой проблемы. Во-первых, вы можете просто заменить кавычки на одинарные.

Другой, более распространенный подход - это код print("hello, \"friend\"). Обратные косые черты указывают компилятору, что следующий символ следует считать кавычками в строке, что позволяет избежать ошибки компилятора.

Форматирование строк

Вероятно, самым элегантным способом использования строк будет следующий:

```
# Запросить у пользователя его имя
name = input("Как вас зовут? ")
print(f "hello, {name}")
```

Обратите внимание на букву f в print(f "hello, {name}"). Это f является специальным указателем для Python, чтобы обработать эту строку особым образом, отличным от предыдущих подходов, которые мы иллюстрировали в этой лекции. Ожидайте, что в этом курсе вы будете использовать этот стиль строк довольно часто.

Подробнее о строках

Вы никогда не должны ожидать, что ваш пользователь будет работать так, как задумано. Поэтому вам необходимо обеспечить коррекцию или проверку вводимых пользователем данных.

Оказывается, в строках встроена возможность удаления пробелов из строки.

Если использовать метод strip для name в виде name = name.strip(), то он удалит все пробелы слева и справа от вводимых пользователем данных. Вы можете изменить свой код следующим образом:

```
# Спросите пользователя о его имени
name = input("Как вас зовут? ")

# Удалите пробелы из строки
name = name.strip()

# Вывести результат
print(f "hello, {name}")
```

При повторном запуске этой программы, независимо от того, сколько пробелов вы введете до или после имени, она удалит все пробелы.

Используя метод заголовка, она выведет имя пользователя в заглавном регистре:

```
# Спросите пользователя о его имени
name = input("Как вас зовут? ")

# Удалите пробельные символы из строки
name = name.strip()

# Заглавная буква каждого слова
name = name.title()

# Выведите результат
print(f "hello, {name}")
```

К этому моменту вы, возможно, очень устали от многократного набора python в окне терминала. Вы можете воспользоваться стрелкой вверх на клавиатуре, чтобы вспомнить последние команды терминала, которые вы выполняли.

Заметьте, что вы можете изменить свой код, чтобы сделать его более эффективным:

```
# Запросить у пользователя его имя
name = input("Как вас зовут? ")

# Удалите пробелы из строки и выделите первую букву каждого слова заглавными буквами
name = name.strip().title()

# Выведите результат
print(f "hello, {name}")
```

Это создает тот же результат, что и предыдущий код.

Мы можем даже пойти дальше!

Попросите пользователя назвать свое имя, удалите пробелы из строки и выделите первую букву каждого слова заглавными буквами

name = input("Как вас зовут? ").strip().title()

Выведите результат

print(f "hello, {name}")

Вы можете узнать больше о строках в документации Python по str

Целые числа или int

В Python целое число называется int.

В мире математики мы знакомы с операторами +, -, *, / и %. Последний оператор % или оператор modulo может быть вам не очень знаком.

Для выполнения кода Python не обязательно использовать окно текстового редактора в компиляторе. Внизу, в терминале, вы можете запустить один только python. В окне терминала вам будет показано >>>. Затем вы можете запустить живой, интерактивный код. Вы можете ввести 1+1, и он выполнит это вычисление. Этот режим не будет часто использоваться в данном курсе.

Снова открыв VS Code, мы можем набрать в терминале code calculator.py. Это создаст новый файл, в котором мы создадим наш собственный калькулятор.

Сначала мы можем объявить несколько переменных.

x = 1

y = 2

z = x + y

print(z)

Естественно, когда мы запускаем python calculator.py, мы получаем результат в окне терминала 3. Мы можем сделать это более интерактивным, используя функцию ввода.Запустив эту программу, мы обнаруживаем, что выходные данные неверны как 12. Почему это может быть?

Ранее мы видели, как +знак объединяет две строки. Поскольку ваш ввод с клавиатуры вашего компьютера поступает в компилятор в виде текста, он обрабатывается как строка. Поэтому нам необходимо преобразовать

этот ввод из строки в целое число. Мы можем сделать это следующим образом:

```
x = input("What's x? ")
y = input("What's y? ")

z = int(x) + int(y)

print(z)
```

Теперь результат правильный. Использование int(x), называется “приведением”, когда значение временно изменяется с одного типа переменной (в данном случае строки) на другой (здесь целое число).

Мы можем улучшить нашу программу следующим образом:

```
x = int(input("What's x? "))
y = int(input("What's y? "))

print(x + y)
```

Это иллюстрирует, что вы можете запускать функции на функциях. Сначала выполняется самая внутренняя функция, а затем выполняется внешняя. Сначала input выполняется функция. Затем int функция.

Вы можете узнать больше в документации Python о int.

Удобство чтения выигрывает

Принимая решение о вашем подходе к задаче кодирования, помните, что можно привести разумные аргументы в пользу многих подходов к одной и той же проблеме.

Независимо от того, какой подход вы применяете к задаче программирования, помните, что ваш код должен быть читаемым. Вы должны использовать комментарии, чтобы дать себе и другим подсказки о том, что делает ваш код. Кроме того, вы должны создавать код таким образом, чтобы его можно было читать.

Основы Float

Значение с плавающей запятой - это действительное число, в котором есть десятичная точка, например 0.52.

Вы можете изменить свой код для поддержки чисел с плавающей запятой следующим образом:

```
x = float(input("What's x? "))
y = float(input("What's y? "))

print(x + y)
```

Это изменение позволяет вашему пользователю вводить 1.2и 3.4представлять общее количество 4.6.

Давайте представим, однако, что вы хотите округлить сумму до ближайшего целого числа. Просматривая документацию по Pythonround, вы увидите, что доступные аргументыround(number[n, ndigits]). Эти квадратные скобки указывают, что программист может указать что-то необязательное. Поэтому вы могли round(n)бы округлить цифру до ближайшего целого числа. В качестве альтернативы, вы могли бы написать код следующим образом:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Create a rounded result
z = round(x + y)

# Print the result
print(z)
```

The output will be rounded to the nearest integer.

Что, если бы мы хотели отформатировать вывод длинных чисел? Например, вместо того, чтобы видеть1000, вы можете захотеть увидеть 1,000. Вы можете изменить свой код следующим образом:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Create a rounded result
z = round(x + y)

# Print the formatted result
print(f"{z:,}")
```

Хотя это довольно загадочно, это print(f"{z:,}")создает сценарий, в котором выводимые zданные будут включать запятые, где результат может выглядеть как 1,000или 2,500.

Подробнее о поплавках

Как мы можем округлять значения с плавающей запятой? Сначала измените свой код следующим образом:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Calculate the result
z = x / y

# Print the result
print(z)
```

При вводе 2в виде x и 3y результат z0.6666666666, по-видимому, становится бесконечным, как и следовало ожидать.

Давайте представим, что мы хотим округлить это значение, мы могли бы изменить наш код следующим образом:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Calculate the result and round
z = round(x / y, 2)

# Print the result
print(z)
```

Как и следовало ожидать, результат будет округлен до ближайших двух десятичных знаков.

Мы также могли бы использовать fstringдля форматирования выходных данных следующим образом:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))

# Calculate the result
z = x / y

# Print the result
print(f"{z:.2f}")
```

Этот зашифрованный f string код отображается так же, как и наша предыдущая стратегия округления.

Вы можете узнать больше в документации Python о float.

Def

Не было бы неплохо создать наши собственные функции?

Давайте вернем наш окончательный код hello.py, введя code hello.pyего в окно терминала. Ваш начальный код должен выглядеть следующим образом:

```
# Ask the user for their name, remove whitespace from the str and capitalize the first letter of each word
name = input("What's your name? ").strip().title()

# Print the output
print(f"hello, {name}")
```

Мы можем улучшить наш код, чтобы создать нашу собственную специальную функцию, которая говорит нам “привет”!

Удалив весь наш код в нашем текстовом редакторе, давайте начнем с нуля:

```
name = input("What's your name? ")
hello()
print(name)
```

При попытке запустить этот код ваш компилятор выдаст ошибку. В конце концов, не существует определенной функции для hello.

Мы можем создать нашу собственную функцию, вызываемую hellosледующим образом:

```
def hello():
    print("hello")



name = input("What's your name? ")
hello()
print(name)
```

Обратите внимание, что все, что находится нижеdef hello(), имеет отступ. Python - это язык с отступами. Он использует отступы, чтобы понять, что является частью вышеупомянутой функции. Поэтому все в hellofункции должно иметь отступ. Когда что-то не имеет отступов, оно обрабатывает его так, как будто его нет внутри hellofункции. Запустив python hello.pyв окне терминала, вы увидите, что ваш результат не совсем такой, как вы хотите.

Мы можем еще больше улучшить наш код:

```
# Create our own function
def hello(to):
    print("hello,", to)


# Output using our own function
name = input("What's your name? ")
hello(name)
```

Здесь, в первых строках, вы создаете свою hello функцию. Однако на этот раз вы сообщаете компилятору, что эта функция принимает один параметр: вызываемую переменную to. Поэтому при вызове hello(name)компьютер переходит name в hello функцию как to. Вот как мы передаем значения в функции. Очень полезно! Запустив python hello.py в окне терминала, вы увидите, что результат намного ближе к нашему идеалу, представленному ранее в этой лекции.

Мы можем изменить наш код, чтобы добавить значение по умолчанию вhello:

```
# Create our own function
def hello(to="world"):
    print("hello,", to)
```

```
# Output using our own function
name = input("What's your name? ")
hello(name)

# Output without passing the expected arguments
hello()
```

Протестируйте свой код самостоятельно. Обратите внимание, что первое `hello`будет вести себя так, как вы могли ожидать, а второе `hello`, которому не передается значение, будет по умолчанию выводиться `hello, world`.

Нам не обязательно иметь нашу функцию в начале нашей программы. Мы можем переместить его вниз, но нам нужно сообщить компилятору, что у нас есть `main`функция, и у нас есть отдельная `hello`функция.

```
def main():

    # Output using our own function
    name = input("What's your name? ")
    hello(name)

    # Output without passing the expected arguments
    hello()



# Create our own function
def hello(to="world"):
    print("hello,", to)
```

Однако это само по себе приведет к возникновению ошибок. Если мы запустим`python hello.py`, ничего не произойдет! Причина этого в том, что ничто в этом коде на самом деле не вызывает `main`функцию и не оживляет нашу программу.

Следующая очень небольшая модификация вызовет mainфункцию и вернет нашу программу в рабочее состояние:

```
def main():

    # Output using our own function
    name = input("What's your name? ")
    hello(name)

    # Output without passing the expected arguments
    hello()



# Create our own function
def hello(to="world"):
    print("hello,", to)



main()
```

Возвращаемые значения

Вы можете представить множество сценариев, в которых вам нужно, чтобы функция не только выполняла действие, но и возвращала значение обратно в основную функцию. Например, вместо того, чтобы просто печатать вычислениех + y, вы можете захотеть, чтобы функция возвращала значение этого вычисления обратно в другую часть вашей программы. Эту “обратную передачу” значения мы называем returnзначением.

Возвращаясь к нашему calculator.рукоду, набрав code calculator.py. Сотрите весь код там. Переработайте код следующим образом:

```
def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))
```

```
def square(n):
    return n * n



main()
```

По сути, x передается square. Затем вычисление x * x возвращается обратно к функции main.

Подводя итоги

Благодаря этой единственной лекции вы изучили способности, которые будете использовать бесчисленное количество раз в своих собственных программах. Вы узнали о…