***Что такое строгая динамическая типизация (strong dynamic type system)?***

*Это приём, используемый в языках программирования и языках спецификации, при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной. Таким образом, в различных участках программы одна и та же переменная может принимать значения разных типов.*

*При‌ ‌сильной‌ ‌или‌ ‌строгой‌ ‌типизации,‌ ‌как‌ ‌в‌ ‌Python,‌ ‌язык‌ ‌не‌ ‌позволяет‌ ‌смешивать‌ ‌разные‌ ‌типы‌ ‌—‌ ‌то‌ ‌есть,‌ ‌если‌ ‌вы‌ ‌обозначили‌ ‌переменную‌ ‌как‌ ‌число,‌ ‌то‌ ‌добавить‌ ‌к‌ ‌ней‌ ‌строку‌ ‌уже‌ ‌не‌ ‌получится.*

***Что такое замыкание функции (closure)?***

*Замыкание (closure) — функция, которая находится внутри другой функции и ссылается на переменные объявленные в теле внешней функции (свободные переменные).*

*Внутренняя функция создается каждый раз во время выполнения внешней. Каждый раз при вызове внешней функции происходит создание нового экземпляра внутренней функции, с новыми ссылками на переменные внешней функции.*

*Ссылки на переменные внешней функции действительны внутри вложенной функции до тех пор, пока работает вложенная функция, даже если внешняя функция закончила работу, и переменные вышли из области видимости.*

*Пример замыкания:*

**def** multiply(num1):  
 var = 10  
   
 **def** inner(num2):  
 **return** num1 \* num2  
   
 **return** inner

*Тут замыканием является функция inner. Функция inner использует внутри себя переменную num1 - параметр функции multiply, поэтому переменная num1 будет запомнена, а вот переменная var не используется и запоминатся не будет.*

***Что такое списковые включения (list comprehension)?***

*Списковые включения в Python являются краткими синтаксическими конструкциями. Их можно использовать для создания списков из других списков, применяя функции к каждому элементу в списке. Имеют следующий синтаксис:*

[ < expression > **for** < element > **in** < iterable > **if** < condition >]

*Примеры:*

lst\_1 = [x + 1 **for** x **in** (1, 2, 3)] *# списковое включение, выдаёт [2, 3, 4]*  
lst\_2 = [x **for** x **in** (1, 2, 3) **if** x % 2 == 0] *# списковое включение с фильтром, выдаст [2]*  
lst\_3 = [x + 1 **if** x % 2 == 0 **else** x **for** x **in** (1, 2, 3)] *# выдаст [1, 3, 3]*

***Что такое принцип DRY?***

*Don’t repeat yourself (DRY; с англ. — «не повторяйся») — это принцип разработки программного обеспечения, нацеленный на снижение повторения информации различного рода, особенно в системах со множеством слоёв абстрагирования. Принцип DRY формулируется как: «Каждая часть знания должна иметь единственное, непротиворечивое и авторитетное представление в рамках системы». Он был сформулирован Энди Хантом и Дэйвом Томасом в их книге Программист-прагматик. Они применяли этот принцип к «схемам баз данных, планам тестирования, сборкам программного обеспечения, даже к документации». Когда принцип DRY применяется успешно, изменение единственного элемента системы не требует внесения изменений в другие, логически не связанные элементы. Те элементы, которые логически связаны, изменяются предсказуемо и единообразно.*

***Некоторые знаки в переводе на английский***

*\_ (нижнее подчеркивание) – underscore*

*- (тире) - dash*

*# (решетка) – sharp or hash*

*“” (кавычки) – quotes*

*(), {}, [] (скобки) – round brackets, curly brackets, square brackets*

*, (запятая) – comma*

*: (двоеточие) - colon*

*; (точка с запятой) – semicolon*

***Некоторые термины в переводе на английский***

*взаимодействие – interaction*

*производительность – performance (productivity)*

*остаток от деления – remainder*

*ссылка – reference*

*явный – explicit*

*неявный - implicit*