<question> 101. Instruction for displaying the *first* element of the list:

<variant>list = [ 'January' , 'February' , 'March' ] ,3

<variant>print(list[1] )

<variant>print(list[0] )

<variant>list = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant>max(list)

<question> 102. *Python*. A set of elements in a specific order is indicated by square brackets ([ ]):

<variant> Literal

<variant> Numbers

<variant> String

<variant> List

<variant> Operator

<question> 103. *Python*. List items are accessed based on:

<variant> Literal

<variant> Numbers

<variant> String

<variant> Operator

<variant> Index

<question> 104. *Python*. Method *for adding new element* ***x***to the end of the *lst* list:

<variant> lst.extend(t)

<variant> lst.index(x)

<variant> lst.append (x)

<variant> lst.pop(i)

<variant> lst.insert (i, x)

<question> 105. *Python*. Method *for adding new element* ***x*** to position ***i*** of the *lst* list:

<variant> lst.append (x)

<variant> lst.extend(t)

<variant> lst.index(x)

<variant> lst.insert (i, x)

<variant> lst.pop(i)

<question> 106. *Python*. Method for Determining the first left position of element***x***in the***lst***list:

<variant> lst.insert (i, x)

<variant> lst.index(x)

<variant> lst.append (x)

<variant> lst.extend(t)

<variant> lst.pop(i)

<question> 107. *Python*. Method for Deleting an element with the number ***i*** from the ***lst*** list:

<variant> lst.index(x)

<variant> lst.insert (i, x)

<variant> lst.append (x)

<variant> lst.pop(i)

<variant> lst.extend(t)

<question> 108. *Python*. Method for Deleting the *x* element in the *lst* list in the first position on the left (Метод для Удаления элемента x в списке *lst* в первой слева позиции):

<variant> lst.sort()

<variant> lst.pop(i)

<variant> lst.index(x)

<variant> lst.remove(x)

<variant> lst.insert (i, x)

<question> 109. *Python*. Method for Sorting the list in ascending order (Метод для Сортировки списка по возрастанию)*:*

<variant> lst.reverse()

<variant> lst.pop(i)

<variant> lst.index(x)

<variant> lst.insert (i, x)

<variant> lst.sort()

<question> 110. *Python*. A method that reverses the order of the list (Метод который меняет порядок списка на обратный)*:*

<variant> lst.sort()

<variant> lst.pop(i)

<variant> lst.index(x)

<variant> lst.reverse()

<variant> lst.insert (i, x)

<question> 111. *Python*. A method that returns the number of elements in the list ***lst*** with the value **х** (Метод который возвращает количество элементов в списке ***lst*** с указанным значением)*:*

<variant> lst.sort()

<variant> lst.pop(i)

<variant> lst.index(x)

<variant> lst.count(x)

<variant> lst.insert (i, x)

<question> 112. *Python*. Function for determining the length of a list (Определяется количество элементов списка)*:*

<variant> append()

<variant> insert()

<variant> reverse()

<variant> len()

<variant> remove()

<question> 113. *Python*. Function for combining lists (Функция для объединения списков)*:*

<variant> len()

<variant> insert()

<variant> reverse()

<variant> lst1+lst2

<variant> remove()

<question> 114. *Python*. Method to Remove all elements from the list (Метод для удаления всех элементы из списка)*:*

<variant> lst.sort()

<variant> lst.pop(i)

<variant> lst.index(x)

<variant> lst.clear(x)

<variant> lst.insert (i, x)

<question> 115. *Python*. In the list *lst = ["apple", "banana", "cherry"]* print the *last element* of the list

(В списке lst = ["яблоко", "банан", "вишня"] распечатайте последний элемент списка):

<variant> print (lst [0])

<variant> print (lst [-1])

<variant> print (lst [1])

<variant> print (lst [3])

<variant> print (lst [-2])

<question> 116. *Python*. In the list *lst = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]* print the elements of the list - *"cherry", "orange", "kiwi"*

(В списке lst =["яблоко","банан","вишня","апельсин","киви","дыня","манго"] распечатайте элементы списка - "вишня","апельсин","киви"):

<variant> print (lst [-1])

<variant> print (lst [3:5])

<variant> print (lst [3:6])

<variant> print (lst [2:5])

<variant> print (lst [-2])

<question> 117. *Py*thon. Built-in function that converts object *x* to a *list*:

<variant> float(x)

<variant> tuple(x)

<variant> int(x)

<variant> str (x)

<variant> list(х)

<question> 118. Instructions *for initializing a list* of three-element list containing string values:

<variant> print(lst[0] )

<variant> quarter = [ 'January', 'February', 'March' ]

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> min(lst)

<variant> max(lst)

<question> 119. Instructions *for displaying the first e*lement of a ***list*** (Инструкция для вывода первого элемента списка):

<variant> lst = [ 'January' , 'February' , 'March' ]

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> max(lst)

<variant> print (lst [0] )

<question> 120. Instruction for creating a multidimensional list of two elements, each of which is a list of three elements containing integer values (Инструкция по созданию многомерного списка из двух элементов, каждый из которых представляет собой список из трех элементов, содержащих целочисленные значения):

<variant> lst = [ 'January' , 'February' , 'March' ], 3

<variant> print (lst[0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> print ( '\nTop Left 0,0 :' , lst[1][0] )

<variant> max(lst)

<question> 121. Instructions for printing the values of a two-dimensional list (Инструкция для печати значений двумерного списка):

<variant> list = [ 'January' , 'February' , 'March' ], 3

<variant> print(lst[0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> max(lst)

<question> 122. Instruction *for determining the list item with the highest value in accordance with*

*the alphabetical* ("dictionary") order (Инструкция для определения элемента списка с наибольшим значением в соответствии с алфавитным ("словарным") порядком):

<variant> list = [ 'January' , 'February' , 'March' ], 3

<variant> print(lst[0] )

<variant> print( '\nTop Left 0,0 :' , lst[1] [0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> max(lst)

<question> 123. Instruction *for replacing list* ***i-index*** *item* with value***x*** (Инструкция по замене элемента списка с ***i***-индексом значением **x**):

<variant> lst = [ 'January', 'February', 'March' ] ,3

<variant> print(lst[0] )

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> lst[i]=x

<question> 124. Instruction *for delete an item* with a number ***i*** from the list (Инструкция для удаления элемента с номером i из списка):

<variant> lst = [ 'January' , 'February' , 'March' ] ,3

<variant> print(list[0] )

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> del lst[i]

<question> 125. Instruction *for finding* *a slice of a list* (*среза списка)* containing elements of the **list** with numbers from **i** to **j** with step **k** (Инструкция для нахождения среза списка, содержащего элементы списка с номерами от **i** до **j** с шагом **k)**:

<variant> lst = [ 'January' , 'February' , 'March' ] ,3

<variant> lst [i:j:k]

<variant> print(lst[0] )

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<question> 126. Instructions *for finding the list element with the smallest value* (*с наименьшим значением)* in accordance with the alphabetical ("dictionary") order:

<variant> min(lst)

<variant> list = [ 'January' , 'February' , 'March' ] , 3

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> max(lst)

<question> 127. Instructions *for determining the number of list element* **lst**:

<variant> len (lst)

<variant> list = [ 'January' , 'February' , 'March' ] , 3

<variant> min(lst)

<variant> print( '\nTop Left 0,0 :' , lst[1][0] )

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<question> 128. Method of removing an element with an i-index from the lst:

<variant> lst = [ 'January' , 'February' , 'March' ] ,3

<variant> colors-tuple = ( 'Red' , 'Green' , 'Red' , 'Blue' , 'Red' )

<variant> lst.pop(i)

<variant> phonetic-set = { 'Alpha' , 'Bravo' , 'Charlie' }

<variant> lst = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<question> 129. The result of the program:



<variant> [2]

<variant> [55]

<variant> [37]

<variant> [20, 61, 2, 37, 4, 55, 36, 7, 18, 39]

<variant> [37, 4, 55]

<question> 130. The result of the program:



<variant> [55]

<variant> [20, 61, 2]

<variant> [37, 4, 55]

<variant> [20, 61, 2, 37, 4, 55, 36, 7, 18, 39]

<variant> 10

<question> 131. The result of the program:



<variant> [2]

<variant> [37]

<variant> [4]

<variant> [55]

<variant> [37, 4, 55]

<question> 132. The result of the program:



<variant> [2]

<variant> [20, 61]

<variant> [37]

<variant> [4]

<variant> [37, 4, 55]

<question> 133. The result of the program:



<variant> [2]

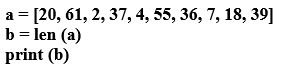
<variant> [37]

<variant> [37, 4, 55]

<variant> [55]

<variant> [20, 61, 2]

<question> 134. The result of the program:



<variant> [37]

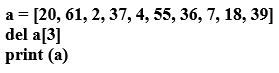
<variant> [4]

<variant> [55]

<variant> 10

<variant> 9

<question> 135. The result of the program:



<variant> [2]

<variant> [55]

<variant> [20, 61, 2, 37, 4, 55, 36, 7, 18, 39]

<variant> [20, 61, 2, 4, 55, 36, 7, 18, 39]

<variant> [37, 4, 55]

<question> 136. The result of the program:



<variant> 2

<variant> [4]

<variant> 61

<variant> [37, 4, 55]

<variant> 10

<question> 137. The result of the program:



<variant> 61

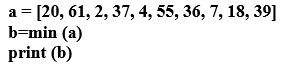
<variant> [4]

<variant> [37, 4, 55]

<variant> 10

<variant> 2

<question> 138. The result of the program:



<variant> 61

<variant> [4]

<variant> 2

<variant> [55]

<variant> 10

<question> 139. Operation ***n-fold*** (n-кратного) list repetition ***lst*:**

<variant> lst ∗ n

<variant> len (lst)

<variant> lst [i:j:k]

<variant> min (lst)

<variant> del lst [i:j]

<question> 140. Operation to find a slice (среза) of a list ***lst*:**

<variant> len (lst)

<variant> lst ∗ n

<variant> del lst [i]

<variant> lst [i:j:k]

<variant> del lst [i:j]

<question> 141. Operation to find the elementwith the smallest value (с наименьшим значением) in the ***lst*:**

<variant> len (lst)

<variant> lst ∗ n

<variant> min (lst)

<variant> max (lst)

<variant> del lst [i:j]

<question> 142. The operation to find the elementwith the highest value (с наибольшим значением) in the list **lst:**

<variant> len (lst)

<variant> lst ∗ n

<variant> min (lst)

<variant> max (lst)

<variant> del lst [i:j]

<question>143. Operation to delete element**i**-indexfrom the list **lst:**

<variant> len (lst)

<variant> lst [i:j:k]

<variant> del lst [i]

<variant> max (lst)

<variant> del lst [i:j]

<question> 144. The operation to delete the elements of the **lst** list included in the slice (срез):

<variant> len (lst)

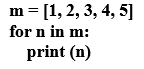
<variant> lst [i:j:k]

<variant> del lst [i:j]

<variant> max (lst)

<variant> del lst [i]

<question> 145. The values of the variable n when executing the loop:



<variant> от 0 до 5

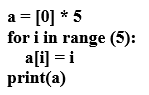
<variant> от 0 до 4

<variant> от 1 до 4

<variant> от 1 до 5

<variant> от 2 до 5

<question> 146. The result of the program:



<variant> b = {2, 4, 6, 8}

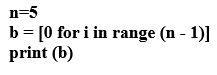
<variant> a = [0, 1, 2, 3]

<variant> a = [0, 1, 2, 3, 4]

<variant> 5, 4, 3, 2, 1

<variant> 1 2 4 6 8

<question> 147. The result of the program:



<variant> {2, 4, 6, 8}

<variant> [0, 1, 2, 3]

<variant> [1, 2, 3, 4]

<variant> [0, 0, 0, 0]

<variant> 1 2 4 6 8

<question> 148. The result of the program:



<variant> {2, 4, 6, 8}

<variant> [0, 1, 2, 3]

<variant> [0, 0, 0, 0]

<variant> ['П', 'Р', 'И', 'В', 'Е', 'Т', '!']

<variant> 1 2 4 6 8

<question> 149. The result of the program:



<variant> {2, 4, 6, 8}

<variant> [0, 0, 0, 0]

<variant> ['П', 'Р', 'И', 'В', 'Е', 'Т', '!']

<variant> [2, 3]

<variant> [0, 1, 2, 3]

<question>150. A block of code that runs only when called to solve a specific task (Блок кода, который запускается только при вызове для решения конкретной задачи):

<variant>Выражение (expression)

<variant>Выявление ошибки (raise an error)

<variant>Функция (Function)

<variant>Глобальный код (global code)

<variant>Идентификатор (identifier)

<question> 151. Definition of the ***Function*** (Определение функции):

<variant> tuple

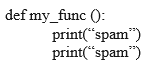
<variant> set

<variant> def

<variant> dictionary

<variant> boolean

<question> 152. Function call (Вызов функции):



<variant> my\_function()

<variant> my\_function('Emil')

<variant> my\_func ()

<variant> my\_func1(“python”)

<variant> func ()

<question> 153. Calling a function with arguments (Вызов функции с аргументами):



<variant> my\_func ()

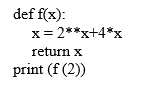
<variant> my\_function('Emil')

<variant> my\_function()

<variant> my\_func1(“python”)

<variant> func ()

<question> 154. Function result (Результат функции):



<variant> 8

<variant> 4

<variant> 12

<variant> 16

<variant> 88

<question> 155. Local variable of the Function (Локальная переменная функции):

<variant> The value of the variable is assigned inside the functions (Значение переменной присваивается внутри функций)

<variant> A variable created outside of a function (if it is visible throughout the program) (Переменная, создаваемая вне функции (если она видна во всей программе))

<variant> Positional arguments (Позиционные аргументы)

<variant> Named arguments (Именованные аргументы)

<variant> Argument (Аргумент)

<question> 156. Global variable of the Function (Глобальная переменная функции):

<variant> The value of the variable is assigned inside the functions (Значение переменной присваивается внутри функций)

<variant> Positional arguments (Позиционные аргументы)

<variant> A variable created outside of a function (it is visible throughout the program) (Переменная, создаваемая вне функции (она видима во всей программе))

<variant> Named arguments (Именованные аргументы)

<variant> Argument (Аргумент)

<question> 157. The arguments of the function are specified in the order exactly corresponding to the order in which the parameters are written (Аргументы функции перечисляются в порядке, точно соответствующем порядку записи параметров):

<variant> A variable created outside of a function (it is visible throughout the program) (Переменная, создаваемая вне функции (она видима во всей программе))

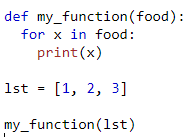
<variant> Positional arguments (Позиционные аргументы)

<variant> The value of the variable is assigned inside the functions (Значение переменной присваивается внутри функций)

<variant> Named arguments (Именованные аргументы)

<variant> Argument (Аргумент)

<question> 158. The result of a function with a list as an named argument (Результат функции со списком в качестве именованного аргумента):



<variant> 6

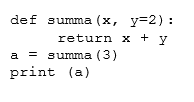
<variant> 12

<variant> 1 2 3

<variant> 10

<variant> 5

<question> 159. The result of a function with a default parameter (Результат функции с параметром по умолчанию):



<variant> 3

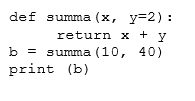
<variant> 2

<variant> 5

<variant> 4

<variant> 6

<question> 160. The result of a function with a default parameter (Результат функции с параметром по умолчанию):



<variant> 12

<variant> 42

<variant> 50

<variant> 4

<variant> 62

<question> 161. Operator for returning function values (Оператор для возвращения значений функции):

<variant> def

<variant> tuple

<variant> return

<variant> global

<variant> dictionary

<question> 162. Definition of an anonymous function (Определение анонимной функции):

<variant> return

<variant> def

<variant> lambda

<variant> global

<variant> dictionary

<question> 163. The result of the function:



<variant> 5

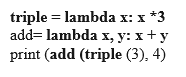
<variant> 6

<variant> 11

<variant> 30

<variant> 35

<question> 164. The result of the function (Результат работы функции):



<variant> 12

<variant> 6

<variant> 11

<variant> 13

<variant> 35

<question> 165. Instructions for creating (packing- упаковки) a tuple:

<variant> list = [ 'January' , 'February' , 'March' ] ,3

<variant> colors-tuple = ('Red' , 'Green' , 'Red' , 'Blue' , 'Red' )

<variant> print( '\nTop Left 0,0 :' , list[1][0] )

<variant> list = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> len (list)

<question> 166. Python. An ordered and immutable collection of data (Упорядоченная и неизменяемая коллекция данных):

<variant> dictionary

<variant> string

<variant> turple

<variant> float

<variant> operator

<question> 167. Python. Data type – tuple (кортеж):

<variant> key()

<variant> del

<variant> tuple()

<variant> pop()

<variant> remove()

<question> 168. Tuple of Python:

<variant> a= 154

<variant> a= ‘Red’

<variant> a = ( 'Red' , 'Green' , 'Red' , 'Blue' , 'Red' )

<variant> a = [20, 61, 2, 37, 4, 55, 36, 7, 18, 39]

<variant> a= { 'Alpha' , 'Bravo' , 'Charlie' }

<question> 169. Definition (packaging) of a tuple:

<variant> pr = (200, 50)

<variant> ‘color’

<variant> s1 = {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<variant> [2, 3]

<variant> s1 = set ([3,6,3,5])

<question> 170. Python. Program result:



<variant> [50, 10, 2]

<variant> {50, 10}

<variant> (50, 10)

<variant> (50, 10, 2)

<variant> (12, 50, 10, 2)

<question> 171. Python. Program result:



<variant> [50, 10, 2, 4]

<variant> {50, 10}

<variant> (50, 10)

<variant> (50, 10, 2)

<variant> (10, 2, 4, 15)

<question> 172. Python. Insert the required operator to print out all the elements of the tuple (вставьте нужный оператор для распечатки всех элементов кортежа):



<variant> range (len(a))

<variant> range(1:6)

<variant> len(a)

<variant> range(len(turple))

<variant> (50, 10, 2)

<question> 173. Set of Python:

<variant> a = 154.02

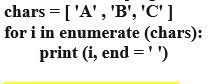
<variant> a = ‘Red’

<variant> a = { 'Alpha' , 'Bravo' , 'Charlie' }

<variant> a = [20, 61, 2, 37, 4, 55, 36, 7, 18, 39]

<variant> a = ( 'Red' , 'Green' , 'Red' , 'Blue' , 'Red' )

<question> 174. The result of the program

****

<variant> A B C

<variant> 0 1 2

<variant> (0, 'A') (1, 'B') (2, 'C')

<variant> 10

<variant> 25

<question> 175. The result of the program:



<variant> {2, 4, 6, 8}

<variant> [2, 3]

<variant> {3, 5, 6}

<variant> ['П', 'Р', 'И', 'В', 'Е', 'Т', '!']

<variant> {3, 6, 3, 5}

<question> 176. Instructions *for creating a set*:

<variant> list = [ 'January' , 'February' , 'March' ] ,3

<variant> colors-tuple = ( 'Red' , 'Green' , 'Red' , 'Blue' , 'Red' )

<variant> phonetic-set = { 'Alpha' , 'Bravo' , 'Charlie' }

<variant> list = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

<variant> len (list)

<question> 177. A set method that adds an element **x** to a **set**:

<variant> set.update (*x*, *y*, *z*)

<variant> set.pop ()

<variant> set.add (x)

<variant> set1.intersection(*set2)*

<variant> set1.difference(*set2*)

<question> 178. A set method that *adds multiple elements* to a set:

<variant> set.add (*x*)

<variant> set.update (x, y, z)

<variant> set.pop ()

<variant> set1.difference(*set2*)

<variant> *set1*.union(*set2*)

<question> 179. A set method that *removes one element from a set at random*:

<variant> set.add (*x*)

<variant> set.update (*x*, *y*, *z*)

<variant> set.pop ()

<variant> *set1*.union(*set2*)

<variant> set.discard( *i* )

<question> 180. A set method that returns elements belonging to both sets **set1** and **set2** (пересечение множеств **set1** & **set2**):

<variant> set.add (*x*)

<variant> set.update (*x*, *y*, *z*)

<variant> set1.difference (*set2*)

<variant> set1.intersection (set2)

<variant> set1.union(*set2*)

<question> 181. A set method that returns elements from the set **set1** that are not in **set2** (разность множеств **set1** - **set2**):

<variant> set.add (*x*)

<variant> set.update (*x*, *y*, *z*)

<variant> set.pop ()

<variant> set1.difference (set2)

<variant> set1.intersection (*set2)*

<question> 182. Definition of a ***set***:

<variant> ‘color’

<variant> pr = (200, 50)

<variant> s2 = {3, 6, 3, 5}

<variant> s1 = {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<variant> [2, 3]

<question> 183. The result of the program:



<variant> ‘color’

<variant> (200, 50)

<variant> {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

<variant> [2, 3]

<variant> {3, 6, 3, 5}

<question> 184. The result of the program:



<variant> {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

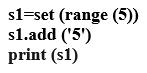
<variant> (200, 50)

<variant> {3, 5, 6, 7}

<variant> {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<variant> {7, 3, 6, 3, 7, 5}

<question> 185. The result of the program:



<variant> {0, 1, 2, 3, 4, 5}

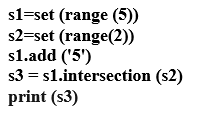
<variant> {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<variant> {0, 1, 2, 3, 4, '5'}

<variant> {3, 5, 6, 7}

<variant> {7, 3, 6, 3, 7, 5}

<question> 186. The result of the program:



<variant> {0, 1, 2, 3, 4, '5'}

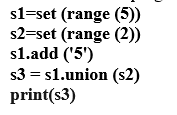
<variant> {0, 1, 2, 3, 4, 5}

<variant> {0, 1}

<variant> (200, 50)

<variant> {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<question> 187. The result of the program:



<variant> {0, 1, 2, 3, 4, 5}

<variant> {0, 1, 2, 3, 4, '5'}

<variant> (200, 50)

<variant> {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<variant> {0, 1}

<question> 188. A set method that returns a new set that includes all elements of ***set1*** and all elements of ***set2*** that are not in ***set1*** (объединение множеств ***set1*** *|* ***set2***):

<variant> set.add (*x*)

<variant> set.update (*x*, *y*, *z*)

<variant> set1.union(set2)

<variant> set.pop ()

<variant> set1.intersection (*set2)*

<question> 189. A container that can contain multiple data elements as a set of pairs **«*key: value*»**:

<variant> dictionary

<variant> turple

<variant> string

<variant> float

<variant> operator

<question> 190. Definition of a ***dictionary***:

<variant> ‘color’

<variant> pr = (200, 50)

<variant> [2, 3]

<variant> d = {'cat': 'кошка', 'dog': 'собака', 'mouse': 'мышь'}

<variant> s2 = {3, 6, 3, 5}

<question> 191. A dictionary method that returns a representation of all pairs (*key, value)* in dictionary ***d***:

<variant> d.keys()

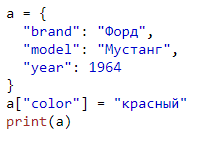
<variant> set.pop ()

<variant> set1.difference (*set2*)

<variant> d.items()

<variant> d.pop(k)

<question> 192. The result of the program: adding an element to the dictionary using a new index key and assigning a value to it (Результат программы: добавление элемента в словарь с помощью нового ключа индекса и присвоения ему значения)



<variant> {'brand': 'Форд', 'model': 'Мустанг', 'year': 1964}

<variant> {'brand': 'Форд', 'model': 'Мустанг', 'year': 1964, 'color': 'красный'}

<variant> ('brand': 'Форд', 'model': 'Мустанг', 'year': 1964)

<variant> ['brand': 'Форд', 'model': 'Мустанг', 'year': 1964, 'color': 'красный']

<variant> {'color': 'красный'}

<question> 193. The method of updating the dictionary with elements from the specified argument (Метод обновления словаря элементами из заданного аргумента)

<variant> pop()

<variant> popitem()

<variant> update()

<variant> del

<variant> clear()

<question> 194. Method for deleting an element from the dictionary with the specified key name (Метод удаления элемента из словаря с указанным именем ключа):

<variant> update()

<variant> popitem()

<variant> pop()

<variant> del

<variant> clear()

<question> 195. A method for clearing the dictionary (Метод для очищения словаря):

<variant> pop()

<variant> clear()

<variant> update()

<variant> popitem()

<variant> del

<question> 196. Method for the method for returning dictionary keys (Метод для возвращения ключей словаря):

<variant> pop()

<variant> keys()

<variant> update()

<variant> popitem()

<variant> del

<question> 197. A method that returns a list of all values in the dictionary (Метод, который возвращает список всех значений в словаре):

<variant> keys()

<variant> pop()

<variant> [values()](https://schoolsw3.com/python/ref_dictionary_values.php)

<variant> update()

<variant> popitem()

<question> 198. A method that returns a dictionary with the specified keys and value (Метод, который возвращает словарь с указанными ключами и значением):

<variant> key()

<variant> del

<variant> [fromkeys()](https://schoolsw3.com/python/ref_dictionary_fromkeys.php)

<variant> pop()

<variant> remove()

<question> 199. Method that returns a copy of the dictionary (Метод, который возвращает копию словаря):

<variant> values()

<variant> key()

<variant> [copy()](https://schoolsw3.com/python/ref_dictionary_copy.php)

<variant> pop()

<variant> remove()

<question> 200. Output the value of the "*model*" key of the ***car*** dictionary using the get method(Вывести значение ключа "*model*" словаря ***car*** используя метод ***get***):



<variant> car.get ("model")

<variant> get("model")

<variant> key. get("model")

<variant> pop(car)

<variant> remove()