

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с функциями в языке Python»

ОТЧЕТ
по лабораторной работе №11
дисциплины
«Основы программной инженерии»

Выполнил:

Гълбачева Доротея Андреева
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

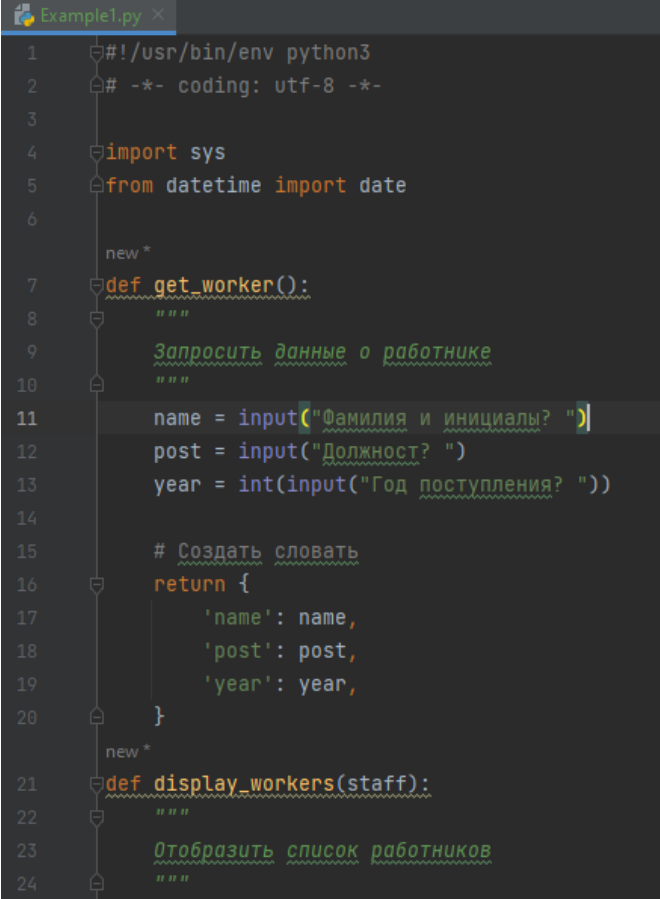
Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Выполнения лабораторной работы:

1. Проработка примеров из лабораторной работы:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  from datetime import date
6
7  new "
8  def get_worker():
9      """
10     Запросить данные о работнике
11     """
12     name = input("Фамилия и инициалы? ")
13     post = input("Должность? ")
14     year = int(input("Год поступления? "))
15
16     # Создать словарь
17     return {
18         'name': name,
19         'post': post,
20         'year': year,
21     }
22
23 new "
24 def display_workers(staff):
25     """
26     Отобразить список работников
27     """
```

Рисунок 11.1 - Код программы примера №1

```

Example1.py x
23     Отобразить список работников
24     """
25     # Проверить, что список работников не пуст
26     if staff:
27         # Заголовок таблицы
28         line = '+-{}-+-{}-+-{}-+-{}+'.format(
29             '-' * 4,
30             '-' * 30,
31             '-' * 20,
32             '-' * 8
33         )
34         print(line)
35         print(
36             '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
37                 "№",
38                 "Ф.И.О.",
39                 "Должность",
40                 "Год"
41             )
42         )
43         print(line)
44
45     # Вывести данные о всех сотрудниках
46     for idx, worker in enumerate(staff, 1):
47         print(
48             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(

```

Рисунок 11.2 - Код программы примера №1

```

Example1.py x
47         print(
48             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
49                 idx,
50                 worker.get('name', ''),
51                 worker.get('post', ''),
52                 worker.get('year', 0)
53             )
54         )
55         print(line)
56     else:
57         print("Список работников пуст.")
58
59     new *
60     def select_workers(staff, period):
61         """
62         Выбирать работников с заданным стажем
63         """
64         # Получить текущую дату
65         today = date.today()
66
67         # Сформировать список работников
68         result = []
69         for employee in staff:
70             if today.year - employee.get('year', today.year) >= period:
71                 result.append(employee)

```

Рисунок 11.3 - Код программы примера №1

```

Example1.py x
72     # Возврат списка выбранных работников
73     return result
74
75     new *
76     def main():
77         """
78         Главная функция программы
79         """
80         # Список работников
81         workers = []
82
83         # Организовать бесконечный цикл запроса команд
84         while True:
85             # Запросить команду из терминала
86             command = input(">>> ").lower()
87
88             # Выполнить действие в соответствие с командой
89             if command == 'exit':
90                 break
91
92             elif command == 'add':
93                 # Запросить данные о работнике
94                 worker = get_worker()
95
96                 # Добавить словарь в список
97                 workers.append(worker)

```

Рисунок 11.4 - Код программы примера №1

```

Example1.py x
93         worker = get_worker()
94
95         # Добавить словарь в список
96         workers.append(worker)
97         # Отсортировать список в случае необходимости
98         if len(workers) > 1:
99             workers.sort(key=lambda item: item.get('name', ''))
100
101         elif command == 'list':
102             # Отобразить всех работников
103             display_workers(workers)
104
105         elif command.startswith('select '):
106             # Разбить команду на части для выделения стажа
107             parts = command.split(' ', maxsplit=1)
108             # Получить требуемый стаж
109             period = int(parts[1])
110
111             # Выбрать работников с заданным стажем
112             selected = select_workers(workers, period)
113             # отобразить выбранных работников
114             display_workers(selected)
115
116         elif command == 'help':
117             # Вывести справку о работе с программой
118             print("Список команд:\n")

```

Рисунок 11.5 - Код программы примера №1

```

103     display_workers(workers)
104
105     elif command.startswith('select '):
106         # Разбить команду на части для выделения стажа
107         parts = command.split(' ', maxsplit=1)
108         # Получить требуемый стаж
109         period = int(parts[1])
110
111         # Выбрать работников с заданным стажем
112         selected = select_workers(workers, period)
113         # отобразить выбранных работников
114         display_workers(selected)
115
116     elif command == 'help':
117         #Вывести справку о работе с программой
118         print("Список команд:\n")
119         print("add - добавить работника;")
120         print("list - вывести список работников;")
121         print("select <стаж> - запросить работников со стажем;")
122         print("help - отобразить справку;")
123         print("exit - завершить работу с программой")
124
125     else:
126         print(f"Неизвестная команда {command}", file=sys.stderr)
127

```

Рисунок 11.6 - Код программы примера №1

```

Example1 >>> help
Список команд:
add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой
>>> add
Фамилия и инициалы? Иванов Д.Н.
Должность? Программист
Год поступления? 2018
>>> add
Фамилия и инициалы? Петров Н.К.
Должность? Геймдизайнер
Год поступления? 2020
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Иванов Д.Н.              | Программист         |  2018   |
|  2 | Петров Н.К.              | Геймдизайнер        |  2020   |
+-----+-----+-----+-----+
>>> select 1
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Иванов Д.Н.              | Программист         |  2018   |
|  2 | Петров Н.К.              | Геймдизайнер        |  2020   |
+-----+-----+-----+-----+
>>> exit

```

Рисунок 11.7 - Результат работы программы пример №1

Задание № 1:

Решить следующую задачу: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции `test()` и инструкции `if __name__ == '__main__':`. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция `positive()`, тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное". Понятно, что вызов `test()` должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения `positive()` и `negative()` предшествовать `test()` или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

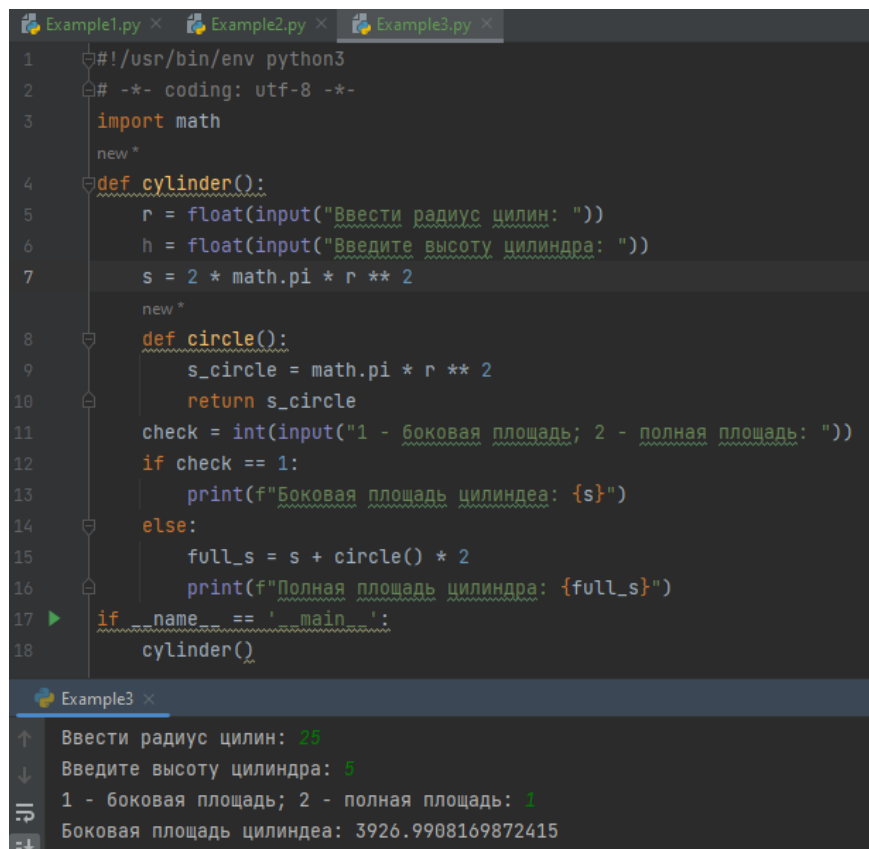


```
Example1.py x Example2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  new *
5  def test():
6      num = int(input("Введите число: "))
7      if num > 0:
8          positive()
9      elif num < 0:
10         negative()
11     else:
12         print("Ваше число равно 0")
13
14  new *
15  def positive():
16     print("Ваше число положительное")
17
18  Example2 x
19  C:\Users\user\LabR_11\venv\Scripts\python.exe
20  Введите число: -13
21  Ваше число отрицательное
```

Рисунок 11.8 - Код и результат работы программы задания №1

Задание №2:

Решите следующую задачу: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле $S = \pi r^2$. В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле $S_{\text{бок}} = 2\pi r h$, или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.



```
1  #!/usr/bin/env python3
2  #- coding: utf-8 -*-
3  import math
4  new *
5  def cylinder():
6      r = float(input("Ввести радиус цилиндра: "))
7      h = float(input("Введите высоту цилиндра: "))
8      s = 2 * math.pi * r ** 2
9      new *
10     def circle():
11         s_circle = math.pi * r ** 2
12         return s_circle
13     check = int(input("1 - боковая площадь; 2 - полная площадь: "))
14     if check == 1:
15         print(f"Боковая площадь цилиндра: {s}")
16     else:
17         full_s = s + circle() * 2
18         print(f"Полная площадь цилиндра: {full_s}")
19 if __name__ == '__main__':
20     cylinder()
```

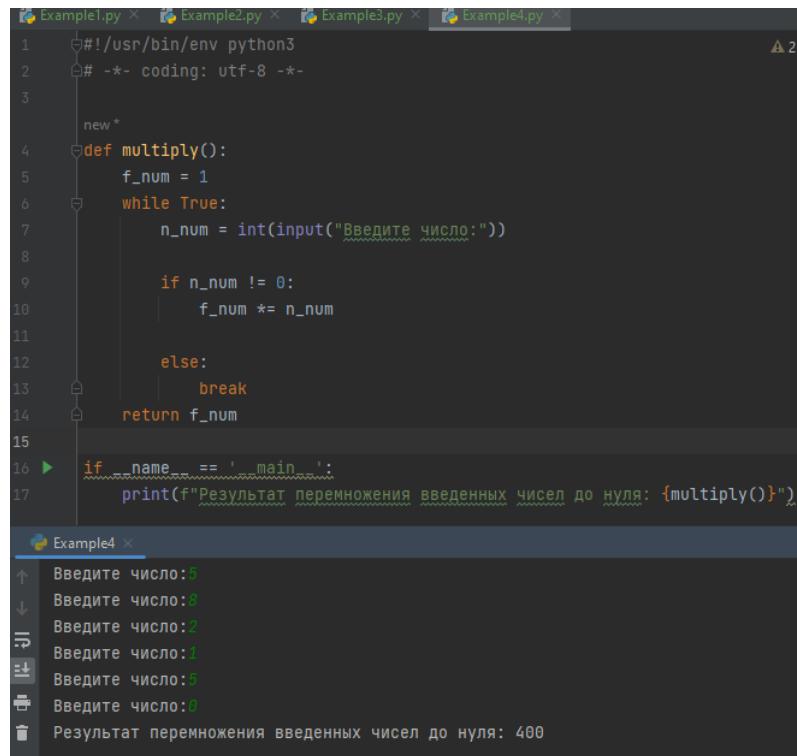
Example3

↑ Ввести радиус цилиндра: 25
↓ Введите высоту цилиндра: 5
1 - боковая площадь; 2 - полная площадь: 1
Боковая площадь цилиндра: 3926.9908169872415

Рисунок 11.9 - Код и результат работы программы задания №2

Задание №3:

Решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.



```
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3
4  new *
5  def multiply():
6      f_num = 1
7      while True:
8          n_num = int(input("Введите число:"))
9          if n_num != 0:
10             f_num *= n_num
11          else:
12             break
13      return f_num
14
15
16 if __name__ == '__main__':
17     print(f"Результат перемножения введенных чисел до нуля: {multiply()}")
```

Example4

Введите число: 5
Введите число: 8
Введите число: 2
Введите число: 1
Введите число: 3
Введите число: 0
Результат перемножения введенных чисел до нуля: 480

Рисунок 11.10 - Код и результат работы программы задания №3

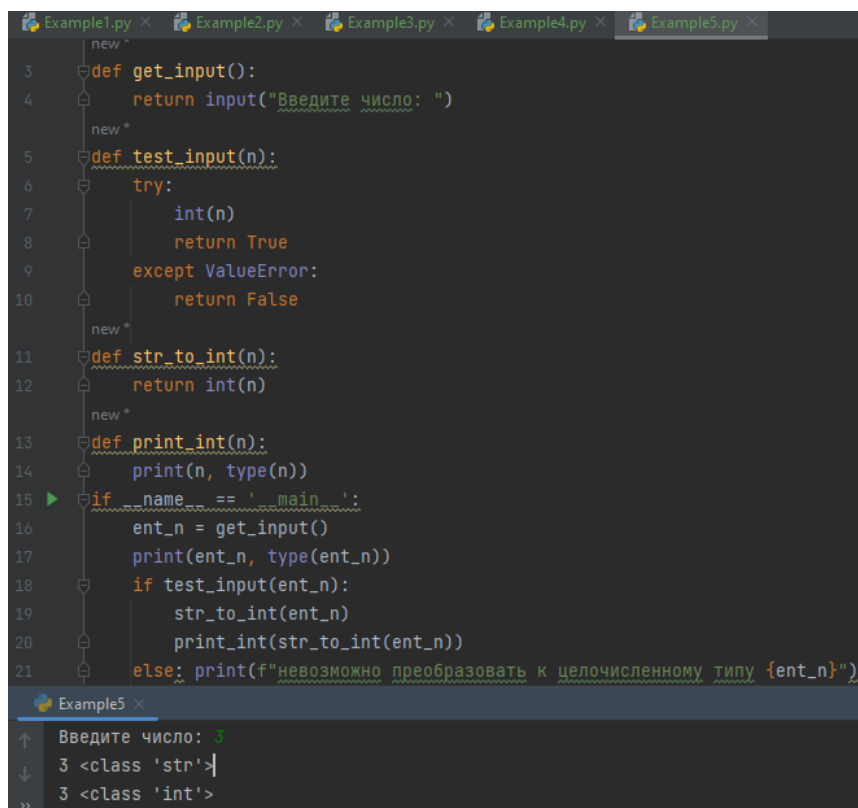
Задание №4:

Решите следующую задачу: напишите программу, в которой определены следующие четыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.
2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.
3. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.
4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция

вернула True, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.



```
3 def get_input():
4     return input("Введите число: ")
5
6 def test_input(n):
7     try:
8         int(n)
9         return True
10    except ValueError:
11        return False
12
13 def str_to_int(n):
14     return int(n)
15
16 def print_int(n):
17     print(n, type(n))
18
19 if __name__ == '__main__':
20     ent_n = get_input()
21     print(ent_n, type(ent_n))
22     if test_input(ent_n):
23         str_to_int(ent_n)
24         print_int(str_to_int(ent_n))
25     else: print(f"невозможно преобразовать к целочисленному типу {ent_n}")
```

Example5

↑ Введите число: 3

↓ 3 <class 'str'>

3 <class 'int'>

Рисунок 11.11 - Код и результат работы программы задания №4

2. Индивидуальные задания:

Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  def add_train():
8      destination = input("Название пункта назначения: ")
9      number = int(input("Номер поезда: "))
10     time = input("Время отправления: ")
11
12     train = {
13         'destination': destination,
14         'number': number,
15         'time': time,
16     }
17
18
19  def list_trains(trains):
20     if trains:
21         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
22             '-' * 4,
23             '-' * 30,
24             '-' * 20,
25             '-' * 8
26         )

```

Рисунок 11.12 - Код программы задания №1

```

27     print(line)
28     print(
29         '| {:>4} | {:<10} | {:<10} |'.format(
30             "№ поезда",
31             "Пункт назначения",
32             "Время отправления"
33         )
34     )
35     print(line)
36
37     for idx, train in enumerate(trains, 1):
38         print(
39             '| {:>4} | {:<10} | {:<10} | {:>8} |'.format(
40                 idx,
41                 train.get('number', 0),
42                 train.get('destination', ''),
43                 train.get('time', '')
44             )
45         )
46         print(line)
47     else:
48         print("Список поездов пуст.")
49
50
51  def select_trains(trains, time):
52

```

Рисунок 11.13 - Код программы задания №1

```

54         print(
55             "Поезде с номером {:>1} отправляется в город {:>1} в {:>1}".format(
56                 trains.get('number', ''),
57                 trains.get('destination', ''),
58                 trains.get('time', '')
59             )
60         )
61
62
63     def main():
64         trains = []
65         while True:
66             command = input(">>> ").lower()
67
68             if command == 'exit':
69                 break
70
71             elif command == 'add':
72                 train = add_train()
73                 trains.append(train)
74                 if len(trains) > 1:
75                     trains.sort(key=lambda item: item.get('number', ''))
76
77             elif command == 'list':
78                 list_trains(trains)
79

```

Рисунок 11.14 - Код программы задания №1

```

80         elif command.startswith('select '):
81             parts = command.split(' ', maxsplit=1)
82             num = int(parts[1])
83             result = []
84             for train in trains:
85                 if num == train.get('number', ''):
86                     print(select_trains(trains))
87
88
89         elif command == 'help':
90             print("Список команд:\n")
91             print("add - добавить поезда;")
92             print("list - вывести список поездов;")
93             print("select <номер поезда> запросить информации о поезде с этим номером ")
94             print("help - отобразить справку")
95             print("exit - завершить работу с программой.")
96
97         else:
98             print(f"Неизвестная команда {command}", file=sys.stderr)
99
100
101     if __name__ == '__main__':
102         main()
103

```

Рисунок 11.15 - Код программы задания №1

```
>>> add
Название пункта назначения: Москва
Номер поезда: 23
Время отправления: 12:30
>>> add
Название пункта назначения: София
Номер поезда: 16
Время отправления: 17:45
>>> list
| 1 | 16 | София | 17:45 |
| 2 | 23 | Москва | 12:30 |
>>> select 23
Поезде с номером 23 отправляется в город Москва в 12:30
>>> exit
```

Рисунок 11.16 - Результат работы программы задания №1

3. Контрольные вопросы:

1. Каково назначение функций в языке программирования Python?

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

2. Каково назначение операторов def и return ?

В языке программирования Python функции определяются с помощью оператора def. Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором return.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

4. Как вернуть несколько значений из функции Python?

```
return side, full  
  
scyl, fcy1 = cylinder()
```

5. Какие существуют способы передачи значений в функцию?

Через параметры, и через ввод, запрашиваемый самой функцией

6. Как задать значение аргументов функции по умолчанию?

```
def cylinder(h, r=1):
```

7. Каково назначение lambda-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые lambda-функции могут быть использованы везде, где требуется функция

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python - достаточно важный аспект, ведь от нее порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. PEP 257 описывает

соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP - стандартизировать структуру строк документации: что они должны в себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис.

9. В чем особенность однострочных и многострочных форм строк документации?

Для согласованности, всегда используйте `"""triple double quotes"""` для строк документации. Используйте `r"""raw triple double quotes"""` , если вы будете использовать обратную косую черту в строке документации. Существует две формы строк документации: однострочная и многострочная.