

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2.2
дисциплины «Программирование на Python»

Выполнил:
Степанов Леонид Викторович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение
средств вычислительной
техники и автоматизирование
систем», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

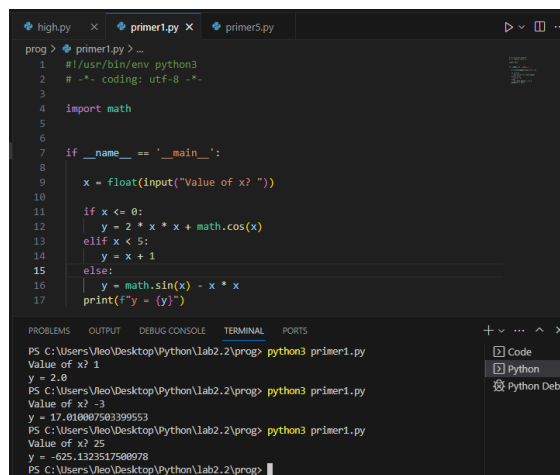
Ставрополь, 2023 г.

Тема: условные операторы и циклы в языке Python

Цель: приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if, while, for, break и continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Порядок выполнения работы:

1. Написал программу (primer1.py), которая вычисляет значение функции при разных входных данных (x), если вводимое значение меньше нуля – вывод равен $2 * x * x * \cos(x)$, иначе если $-x + 1$, иначе $\sin(x) - x * x$



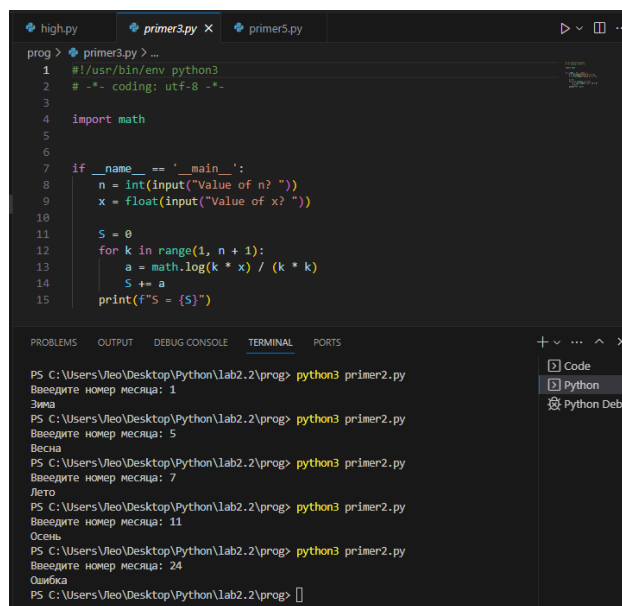
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 if __name__ == '__main__':
8
9     x = float(input("Value of x? "))
10
11     if x <= 0:
12         y = 2 * x * x + math.cos(x)
13     elif x < 5:
14         y = x + 1
15     else:
16         y = math.sin(x) - x * x
17     print(f"y = {y}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer1.py
Value of x? 1
y = 2.0
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer1.py
Value of x? -3
y = 17.010807593399553
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer1.py
Value of x? 25
y = -625.1323517500978
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> |

Рисунок 1 – Результат выполнения программ primer1.py

2. Написал программу (primer2.py), которая выводит время года в зависимости от числа (1-12)



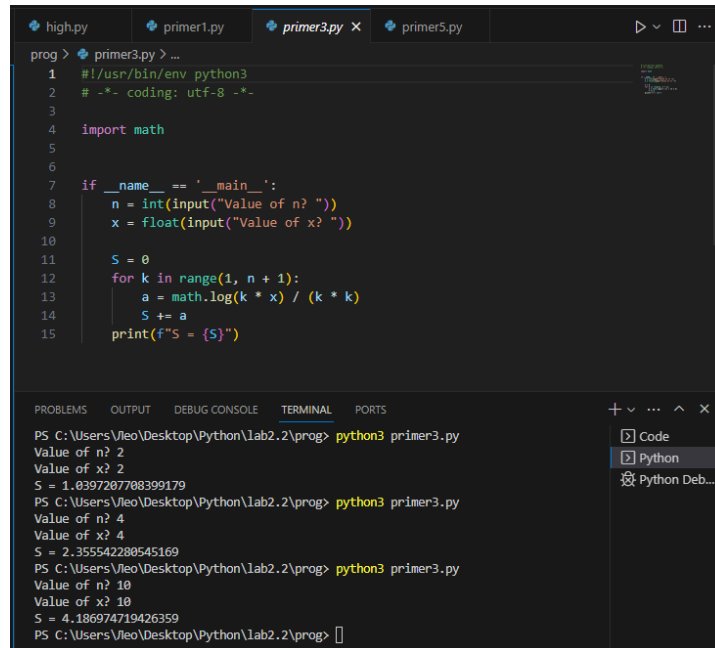
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 if __name__ == '__main__':
8     n = int(input("Value of n? "))
9     x = float(input("Value of x? "))
10
11     S = 0
12     for k in range(1, n + 1):
13         a = math.log(k * x) / (k * k)
14         S += a
15     print(f"S = {S}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer2.py
Введите номер месяца: 1
Зима
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer2.py
Введите номер месяца: 5
Весна
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer2.py
Введите номер месяца: 7
Лето
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer2.py
Введите номер месяца: 11
Осень
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> python3 primer2.py
Введите номер месяца: 24
Ошибка
PS C:\Users\Iveo\Desktop\Python\Lab2.2\prog> |

Рисунок 2 – Результат выполнения программы primer2.py

3. Написал программу (primer3.py), которая вычисляет конечную сумму.

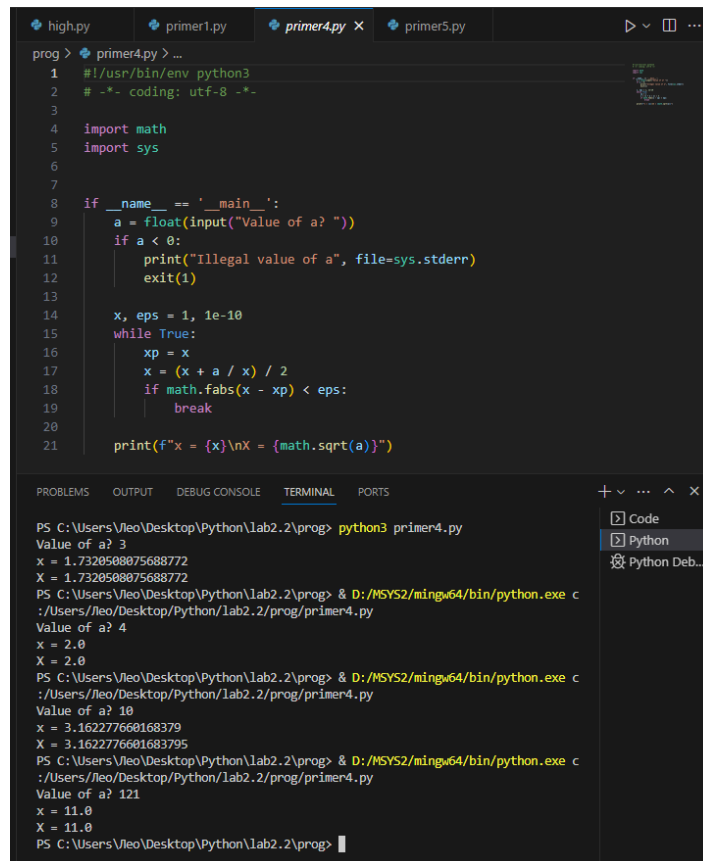


```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 if __name__ == '__main__':
8     n = int(input("Value of n? "))
9     x = float(input("Value of x? "))
10
11     S = 0
12     for k in range(1, n + 1):
13         a = math.log(k * x) / (k * k)
14         S += a
15     print(f"S = {S}")
```

PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 primer3.py
Value of n? 2
Value of x? 2
S = 1.0397207708399179
PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 primer3.py
Value of n? 4
Value of x? 4
S = 2.355542280545169
PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 primer3.py
Value of n? 10
Value of x? 10
S = 4.186974719426359
PS C:\Users\leo\Desktop\Python\lab2.2\prog>

Рисунок 3 – Результат выполнения программы primer3.py

4. Написал программу (primer4.py), которая находит значение квадратного корня из числа с некоторой точностью.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5 import sys
6
7
8 if __name__ == '__main__':
9     a = float(input("Value of a? "))
10     if a < 0:
11         print("Illegal value of a", file=sys.stderr)
12         exit(1)
13
14     x, eps = 1, 1e-10
15     while True:
16         xp = x
17         x = (x + a / x) / 2
18         if math.fabs(x - xp) < eps:
19             break
20
21     print(f"x = {x}\nX = {math.sqrt(a)}")
```

PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 primer4.py
Value of a? 3
x = 1.7320508075688772
X = 1.7320508075688772
PS C:\Users\leo\Desktop\Python\lab2.2\prog> & D:/MSYS2/mingw64/bin/python.exe c
:/Users/leo/Desktop/Python/lab2.2/prog/primer4.py
Value of a? 4
x = 2.0
X = 2.0
PS C:\Users\leo\Desktop\Python\lab2.2\prog> & D:/MSYS2/mingw64/bin/python.exe c
:/Users/leo/Desktop/Python/lab2.2/prog/primer4.py
Value of a? 10
x = 3.162277660168379
X = 3.1622776601683795
PS C:\Users\leo\Desktop\Python\lab2.2\prog> & D:/MSYS2/mingw64/bin/python.exe c
:/Users/leo/Desktop/Python/lab2.2/prog/primer4.py
Value of a? 121
x = 11.0
X = 11.0
PS C:\Users\leo\Desktop\Python\lab2.2\prog>

Рисунок 4 – Результат выполнения программы primer4.py

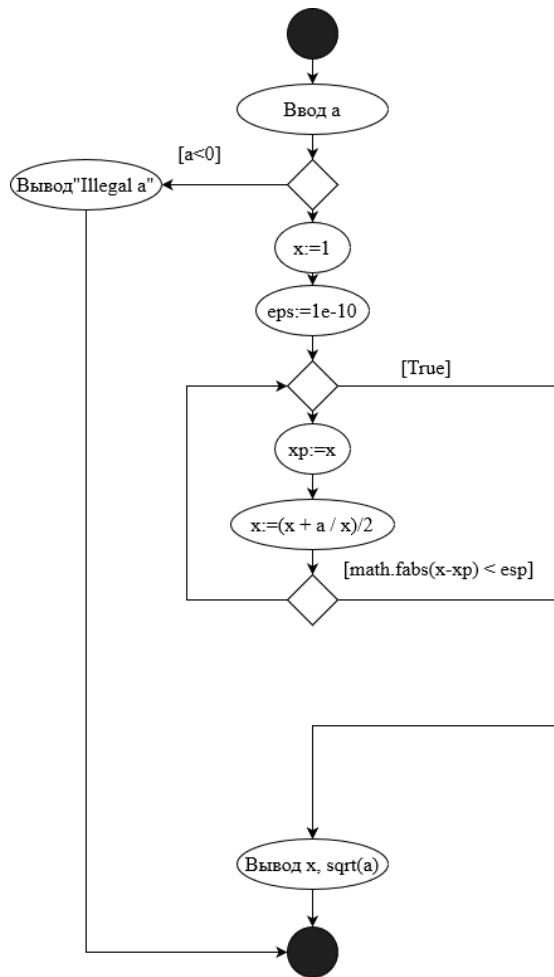


Рисунок 5 – UML-диаграмма primer4.py

5. Написал программу (primer5.py), которая вычисляет значение специальной интегральной показательной функции

```

4 import math
5 from operator import indexOf
6 import sys
7
8 # Постоянная Эйлера.
9 EULER = 0.5772156649015329
10 # Точность вычислений.
11 EPS = 1e-10
12
13 if __name__ == '__main__':
14     x = float(input("Value of x? "))
15     if x == 0:
16         print("Illegal value of x", file=sys.stderr)
17         exit(1)
18
19     a = x
20     S, k = a, 1
21
22     # Найти сумму членов ряда.
23     while math.fabs(a) > EPS:
24         a *= x * k / (k + 1) ** 2
25         S += a
26         k += 1
  
```

```

PS C:\Users\Iveo\Desktop\Python\lab2.2\prog> python3 primer5.py
Value of x? 1
Ei(1.0) = 1.8951178163550635
PS C:\Users\Iveo\Desktop\Python\lab2.2\prog> python3 primer5.py
Value of x? 5
Ei(5.0) = 40.18527535579794
PS C:\Users\Iveo\Desktop\Python\lab2.2\prog> python3 primer5.py
Value of x? 0
Illegal value of x
PS C:\Users\Iveo\Desktop\Python\lab2.2\prog>
  
```

Рисунок 6 – Результат выполнения программы primer6.py

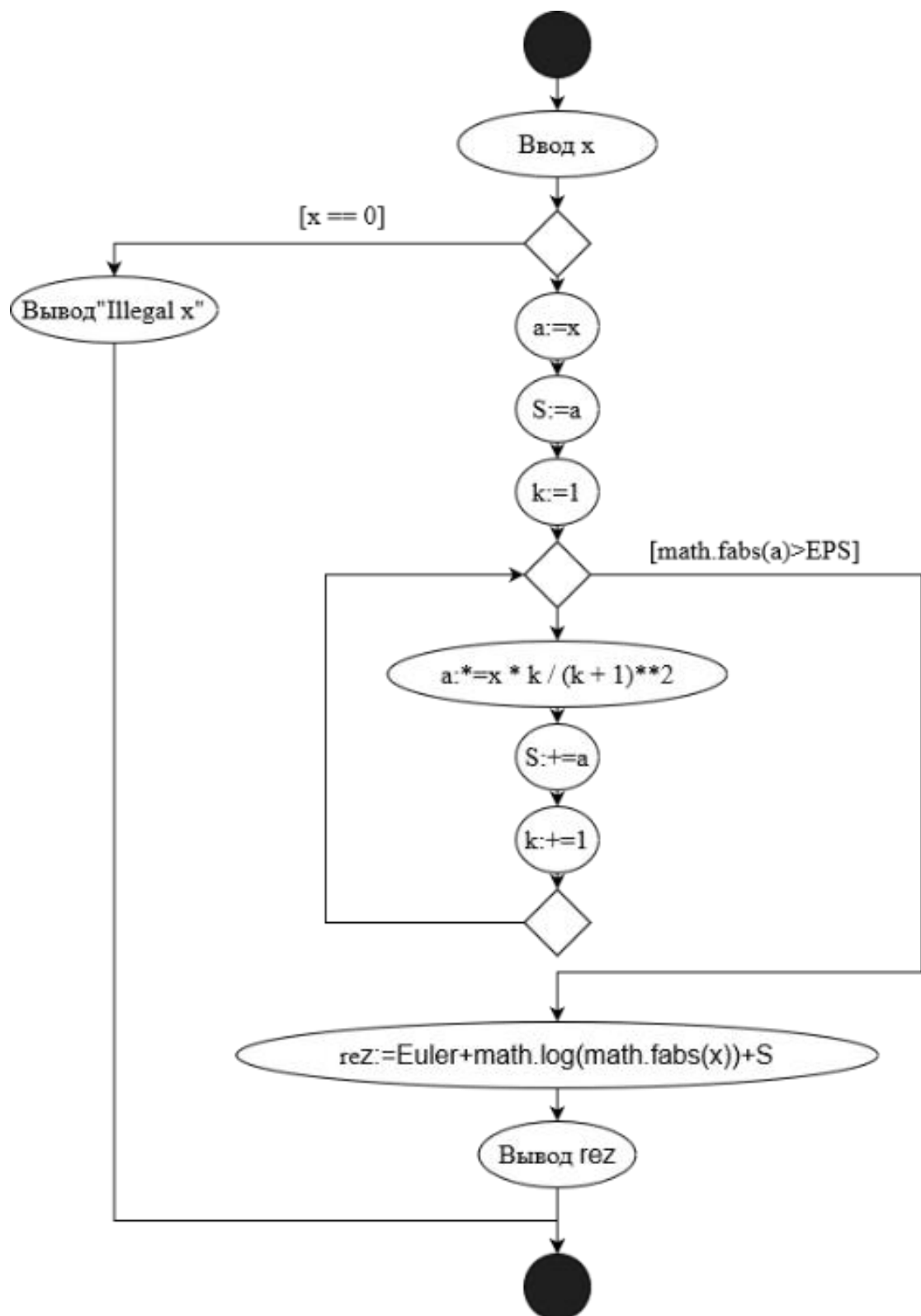


Рисунок 7 – UML-диаграмма primer5.py

Индивидуальные задания (Вариант 17):

1. (В-7) Написал программу (ind1.py), которая выводит время года в зависимости от числа (1-12)

```

high.py  ind2.py  ind1.py  primer1.py  primer5.py
prog > ind1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      n = int(input("Введите номер месяца: "))
9
10     if n == 1 or n == 2 or n == 12:
11         print("Зима")
12     elif n == 3 or n == 4 or n == 5:
13         print("Весна")
14     elif n == 6 or n == 7 or n == 8:
15         print("Лето")
16     elif n == 9 or n == 10 or n == 11:
17         print("Осень")
18     else:
19         print("Ошибка", file=sys.stderr)
20         exit(1)

```

PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind1.py
 Введите номер месяца: 2
 Зима
 PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind1.py
 Введите номер месяца: 6
 Лето
 PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind1.py
 Введите номер месяца: 11
 Осень
 PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind1.py
 Введите номер месяца: 100
 Ошибка

Рисунок 8 – Результат выполнения программы ind1.py

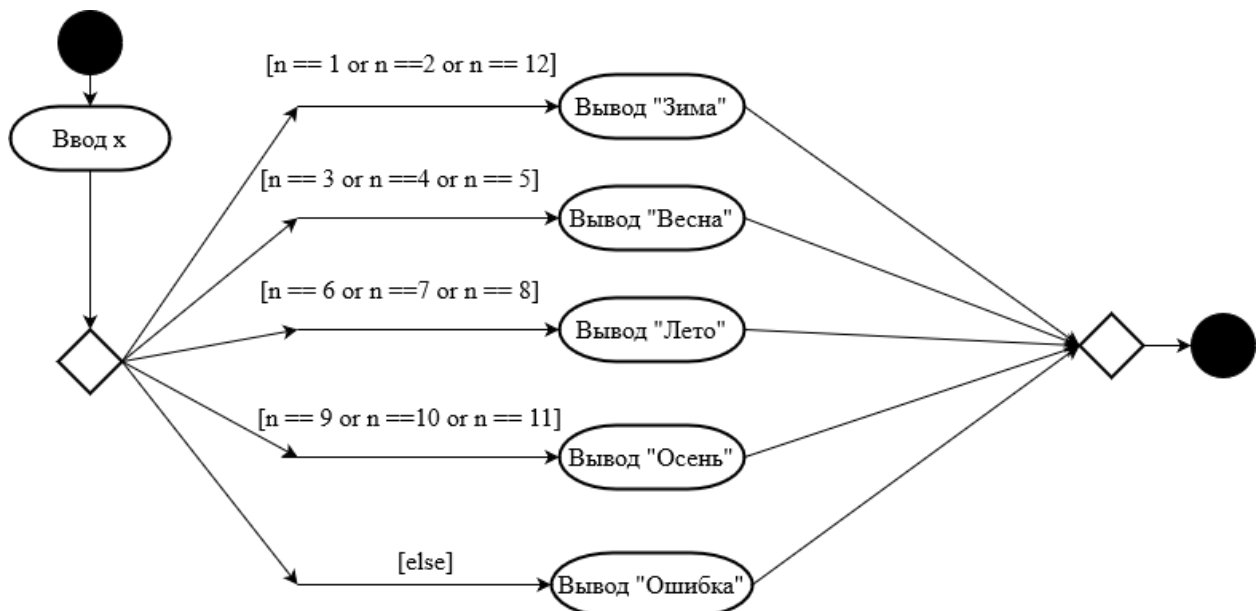


Рисунок 9 – UML-диаграмма ind1.py

2. Написал программу (ind2.py), которая проверяет на четность три введённых числа

```
high.py ind2.py M x ind1.py primer1.py primer5.py
prog > ind2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  if __name__ == '__main__':
6      x = int(input("Введите число: "))
7      y = int(input("Введите число: "))
8      z = int(input("Введите число: "))
9
10     if x % 2 == 0:
11         print(x)
12     if y % 2 == 0:
13         print(y)
14     if z % 2 == 0:
15         print(z)
16
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind2.py
Введите число: 2
Введите число: 3
Введите число: 4
2
4
PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind2.py
Введите число: 333
Введите число: 334
Введите число: 35474
334
35474
PS C:\Users\leo\Desktop\Python\lab2.2\prog>
```

Рисунок 10 – Результат выполнения программы ind2.py

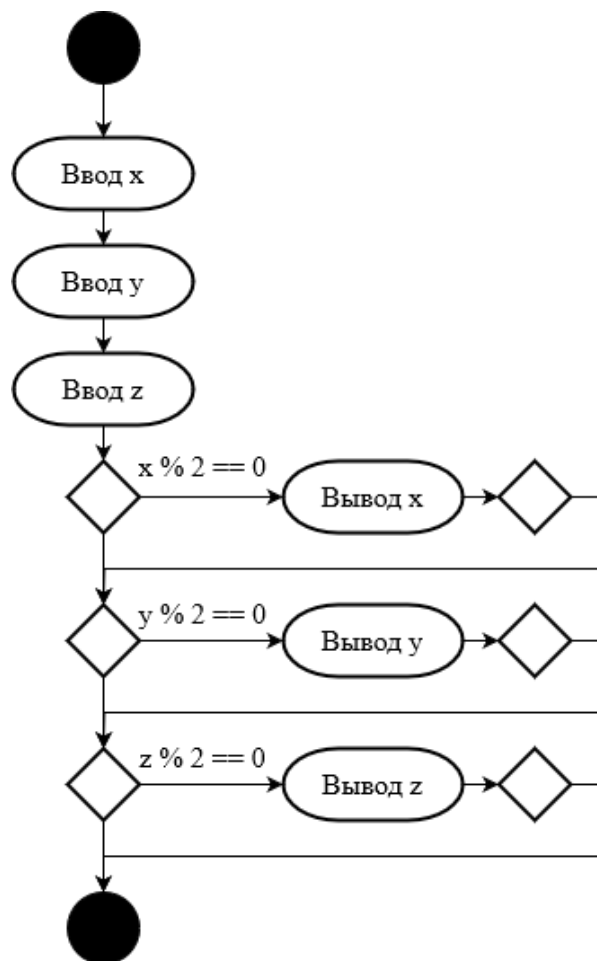
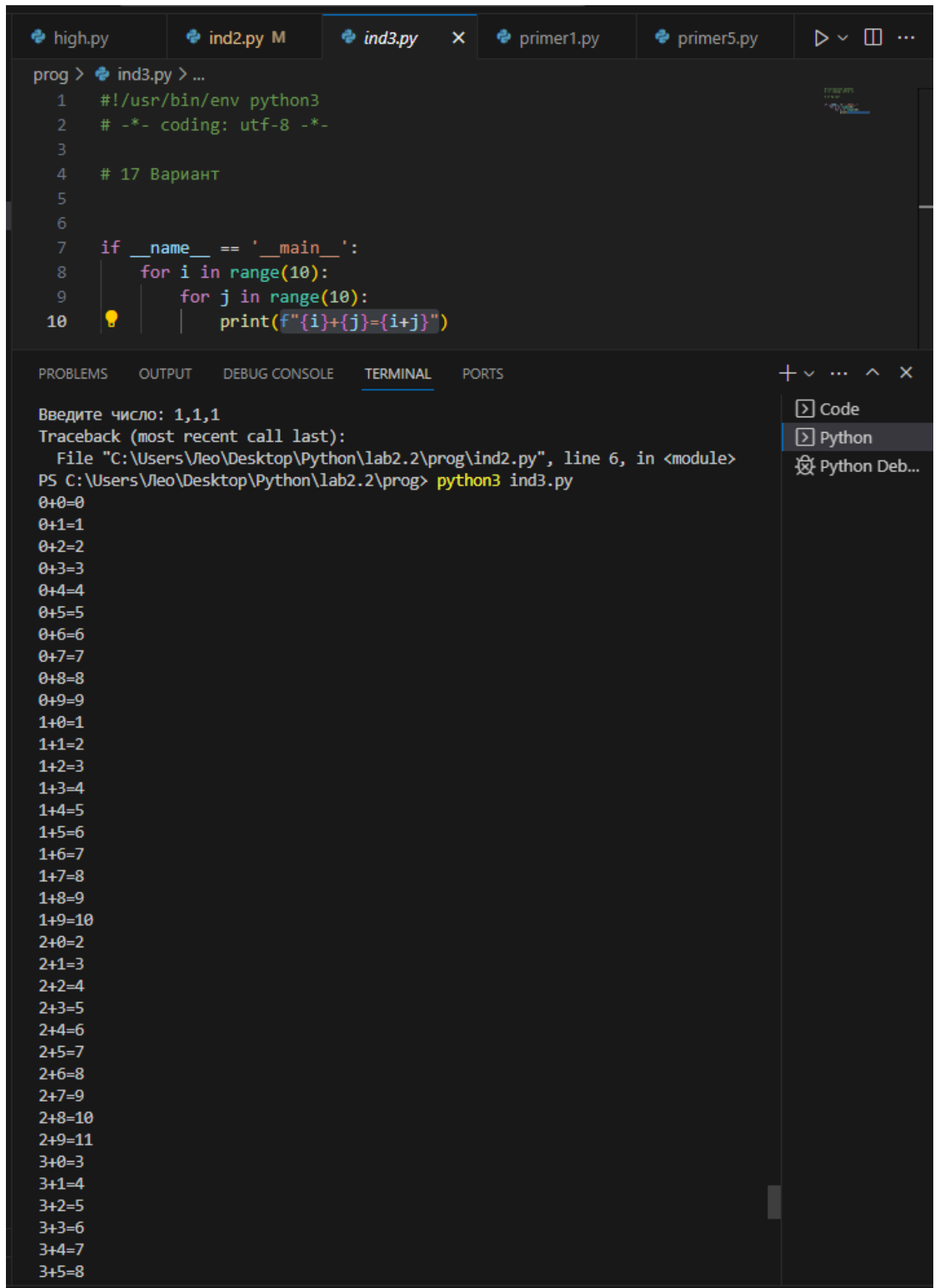


Рисунок 11 – UML-диаграмма ind2.py

3. Написал программу (ind3.py), которая выводит таблицу сумм от 0 до 9.



```
high.py ind2.py M ind3.py x primer1.py primer5.py
prog > ind3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # 17 Вариант
5
6
7  if __name__ == '__main__':
8      for i in range(10):
9          for j in range(10):
10             print(f"{i}+{j}={i+j}")
```

Введите число: 1,1,1
Traceback (most recent call last):
File "C:\Users\leo\Desktop\Python\lab2.2\prog\ind2.py", line 6, in <module>
PS C:\Users\leo\Desktop\Python\lab2.2\prog> python3 ind3.py

0+0=0	0+1=1	0+2=2	0+3=3	0+4=4	0+5=5	0+6=6	0+7=7	0+8=8	0+9=9
1+0=1	1+1=2	1+2=3	1+3=4	1+4=5	1+5=6	1+6=7	1+7=8	1+8=9	1+9=10
2+0=2	2+1=3	2+2=4	2+3=5	2+4=6	2+5=7	2+6=8	2+7=9	2+8=10	2+9=11
3+0=3	3+1=4	3+2=5	3+3=6	3+4=7	3+5=8				

Рисунок 12 – Результат выполнения программы ind3.py

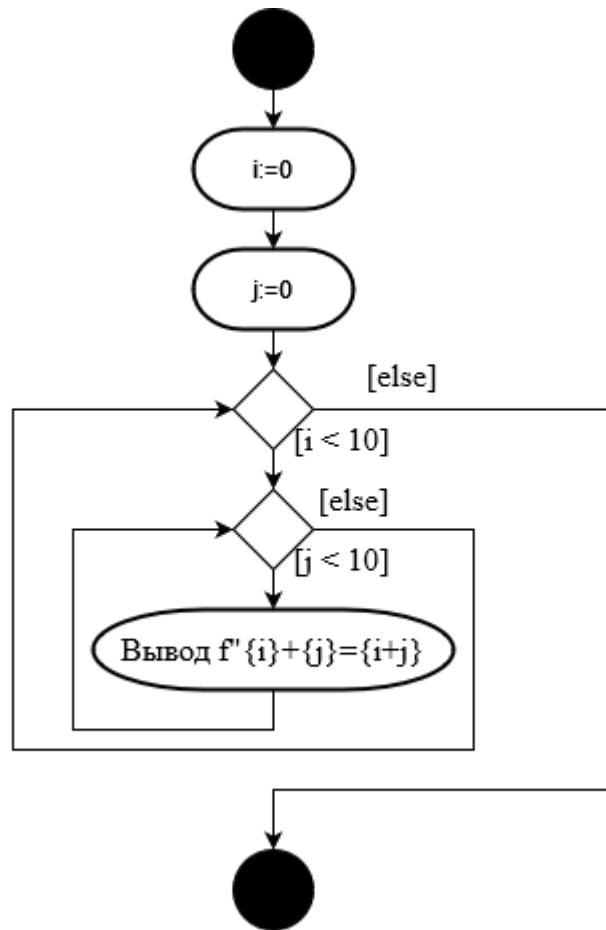


Рисунок 13 – UML-диаграмма ind3.py

Задание повышенной сложности (Вариант 7):

Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью $\epsilon = 10e-10$, аргумент функции вводится с клавиатуры.

7. Функция Бесселя первого рода $I_n(x)$, значение $n = 0, 1, 2, \dots$ также должно вводиться с клавиатуры

$$I_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(x^2/4)^k}{k!(k+n)!} \quad (17)$$

Рисунок 14 – Задание

Прежде чем написать программу я произвёл вычисления отношения общего члена ряда к последующему у меня получилось, что $a_{k+1} =$

$$\frac{x^2}{4k^2 + 4kn + 8k + 4n + 4} a_k$$

Написал программу (high.py), которая вычисляет значение специальной функции, при помощи ранее сделанных вычислений.

```
high.py  X  ind2.py M  ind3.py  primer1.py  primer5.py  ▶ ▢ ...  
prog > high.py > ...  
1  #!/usr/bin/env python3  
2  # -*- coding: utf-8 -*-  
3  
4  import sys  
5  from operator import indexOf  
6  import math  
7  
8  
9  EPS = 1e-10  
10  
11  if __name__ == '__main__':  
12      x = float(input())  
13      n = int(input())  
14  
15      if n < 0:  
16          print("Ошибка", file=sys.stderr)  
17          exit(1)  
18  
19      if x == 0:  
20          print("Ошибка", file=sys.stderr)  
21          exit(1)  
22  
23      a = x  
24      S, k = a, 1  
25  
26      while math.fabs(a) > EPS:  
27          a *= ((x ** 2) / (4 * (k ** 2) + 4 * k * n + 8 * k + 4 * n + 4))  
28          S += a  
29          k += 1  
30  
31      print(f"I({n})({x}) = {S * (x / 2) ** n}")  
32  
  
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
8+9=17  
9+0=9  
9+1=10  
9+2=11  
9+3=12  
9+4=13  
9+5=14  
9+6=15  
9+7=16  
9+8=17  
9+9=18  
PS C:\Users\Leo\Desktop\Python\lab2.2\prog> python3 .\high.py  
1  
1  
I(1)(1.0) = 0.5212728319398761  
PS C:\Users\Leo\Desktop\Python\lab2.2\prog>
```

Рисунок 15 – Результат выполнения программы

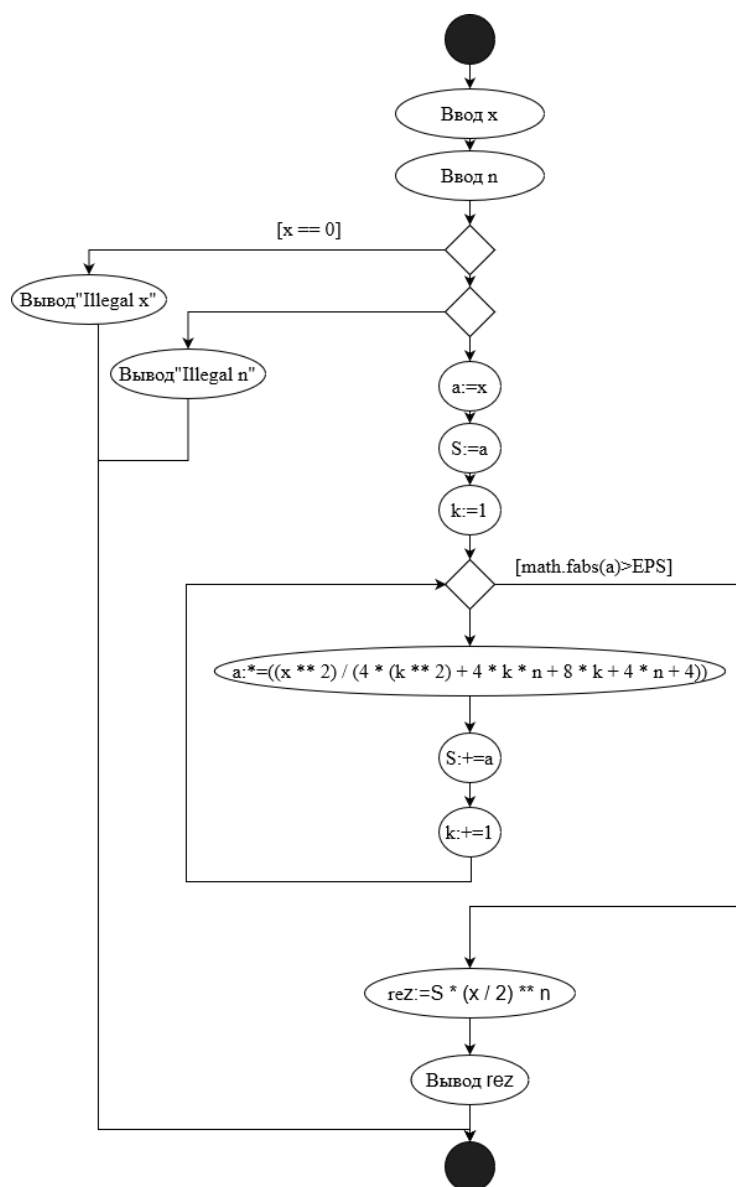


Рисунок 16 – UML-диаграмма ind3.py

Таблица 1 – Результаты выполнения программы

n	x	Результат
1	1,00	0,5212728319
2	1,00	0,2579440744
3	1,00	0,1281687927
4	1,00	0,0638177061
5	1,00	0,0318138892
6	1,00	0,0158714154
7	1,00	0,0079219163
8	1,00	0,0039554500
9	1,00	0,0019754745
1	2,00	2,3625474185
2	2,00	2,2673813724

3	2,00	2,2115180435
4	2,00	2,1748567950
5	2,00	2,1489782253
6	2,00	2,1297475043
7	2,00	2,1149004110
8	2,00	2,1030944173
9	2,00	2,0934836075
1	3,00	6,5423205797
2	3,00	8,9616995274
3	3,00	12,7121161439
4	3,00	18,3628291101
5	3,00	26,8082185550
6	3,00	39,4012230050
7	3,00	58,1711029221
8	3,00	86,1548586342
9	3,00	127,8953441623
1	4,00	15,5189303074
2	4,00	26,5331362517
3	4,00	48,0946186821
4	4,00	89,9530849184
5	4,00	171,4015414415
6	4,00	330,5025913481
7	4,00	642,4241283662
8	4,00	1255,8265335325
9	4,00	2465,1102116584
1	5,00	34,9370274279
2	5,00	69,0269518398
3	5,00	148,3580832477
4	5,00	334,1405373097
5	5,00	774,2433392578
6	5,00	1827,3083173990
7	5,00	4367,0573043733
8	5,00	10530,0266574906
9	5,00	25556,7262289636
1	6,00	77,7892490369
2	6,00	169,1483788690
3	6,00	410,4086447914
4	6,00	1060,9243534238
5	6,00	2852,8645163499
6	6,00	7873,4034863910
7	6,00	22121,9571333507
8	6,00	62957,9512197869
9	6,00	180877,4428069080

1	7,00	174,3303918514
2	7,00	404,1816361627
3	7,00	1070,1209624161
4	7,00	3068,6500271841
5	7,00	9260,7072575055
6	7,00	28923,1660666651
7	7,00	92526,7932203847
8	7,00	301174,9444579630
9	7,00	993037,5812400360
1	8,00	395,8731367826
2	8,00	958,7874945785
3	8,00	2704,9026522335
4	8,00	8392,3649456931
5	8,00	27720,8898928513
6	8,00	95585,5411566619
7	8,00	339807,3401196350
8	8,00	1235281,3563957900
9	8,00	4566313,4828512500
1	9,00	912,3686422373
2	9,00	2278,3231838721
3	9,00	6736,0713799637
4	9,00	22201,6424108958
5	9,00	78752,6318741272
6	9,00	294114,5736255260
7	9,00	1140065,0162785000
8	9,00	4542734,0575155200
9	9,00	18483109,8088863000

Вывод: были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Было освоены операторы языка Python версии 3.x if, while, for, break и continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для визуализации и моделирования последовательности действий и процессов в системе. Они помогают описать, какие действия выполняются, какие ресурсы используются, и какие условия должны быть выполнены для перехода от

одного состояния к другому. Диаграммы деятельности UML широко применяются в разработке программного обеспечения, бизнес-анализе и проектировании бизнес-процессов.

2. Что такое состояние действия и состояние деятельности?

1) Состояние действия в диаграммах деятельности UML представляет конкретное действие или операцию, которую выполняет объект или система в определенный момент времени. Состояние действия может быть представлено в виде прямоугольника с закругленными углами и названием действия внутри.

2) Состояние деятельности в диаграммах деятельности UML представляет набор связанных действий, которые выполняются последовательно или параллельно. Состояние деятельности может быть представлено в виде прямоугольника с закругленными углами и названием деятельности внутри.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

В диаграммах деятельности UML существуют следующие нотации для обозначения переходов и ветвлений:

1) Стрелки с направлением указывают на переходы между состояниями или действиями.

2) Ромбы используются для обозначения ветвлений и объединений. Ветвление происходит, когда одно состояние или действие может привести к нескольким возможным состояниям или действиям. Объединение происходит, когда несколько состояний или действий объединяются в одно состояние или действие.

3) Условия переходов обычно указываются рядом со стрелками или ромбами, чтобы показать, какие условия должны быть выполнены для перехода.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры – это алгоритм, который включает в себя ветвления и условные операторы для принятия решений в

зависимости от определенных условий. Он позволяет программе выполнять различные действия в зависимости от того, выполняется ли определенное условие или нет. Примером алгоритма разветвляющейся структуры может быть условный оператор "if-else" в языке программирования Python.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного алгоритма тем, что разветвляющийся алгоритм содержит ветвления и условные операторы, которые позволяют программе принимать решения и выполнять различные действия в зависимости от определенных условий. Линейный алгоритм, с другой стороны, выполняет последовательные действия без ветвлений или условных операторов. В линейном алгоритме действия выполняются одно за другим в определенном порядке без возможности изменения последовательности выполнения.

6. Что такое условный оператор? Какие существуют его формы?

Условный оператор – это конструкция в программировании, которая позволяет программе принимать решения и выполнять различные действия в зависимости от определенных условий. В языке программирования Python существуют две формы условного оператора: if-else – это форма условного оператора, которая позволяет программе выполнить одно действие, если условие истинно, и другое действие, если условие ложно. if-elif-else – это форма условного оператора, которая позволяет программе проверить несколько условий последовательно и выполнить соответствующее действие для первого истинного условия. Если ни одно из условий не является истинным, выполняется блок кода в разделе else.

7. Какие операторы сравнения используются в Python?

В языке программирования Python используются следующие операторы сравнения:

- 1) == (равно) - проверяет, равны ли два значения.
- 2) != (не равно) - проверяет, не равны ли два значения.
- 3) > (больше) - проверяет, является ли первое значение больше второго.

4) `<` (меньше) - проверяет, является ли первое значение меньше второго.

5) `>=` (больше или равно) - проверяет, является ли первое значение больше или равным второму.

6) `<=` (меньше или равно) - проверяет, является ли первое значение меньше или равным второму.

Эти операторы могут использоваться в условных операторах для сравнения значений и принятия решений.

8. Что называется простым условием? Приведите примеры.

Простое условие – это условие, которое содержит одно сравнение или проверку. В простых условиях используются операторы сравнения для сравнения значений и принятия решений на основе результатов сравнения.

9. Что такое составное условие? Приведите примеры.

Составное условие – это условие, которое состоит из нескольких простых условий, объединенных логическими операторами. Составное условие позволяет программе проверять несколько условий одновременно и принимать решения на основе их комбинации. Примеры составных условий в языке программирования Python:

10. Какие логические операторы допускаются при составлении сложных условий?

При составлении сложных условий в языке программирования Python допускаются следующие логические операторы:

1) `and` - логическое "и". Возвращает `True`, если оба условия истинны.

2) `or` - логическое "или". Возвращает `True`, если хотя бы одно из условий истинно.

3) `not` - логическое "не". Инвертирует значение условия.

Эти операторы позволяют комбинировать простые условия в составные условия и принимать решения на основе их комбинации.

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Да, оператор ветвления в языке программирования Python может содержать внутри себя другие ветвления. Это называется вложенными ветвлениями или вложенными условными операторами. Вложенные ветвления позволяют программе выполнять различные действия в зависимости от нескольких условий, включая проверку условий внутри других условий.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры – это алгоритм, который включает в себя циклы для повторения определенных действий или блоков кода. Циклическая структура позволяет программе выполнять одни и те же действия несколько раз до выполнения определенного условия. Примером алгоритма циклической структуры является цикл "for" или цикл "while" в языке программирования Python.

13. Типы циклов в языке Python.

В языке программирования Python существуют два основных типа циклов:

1) Цикл for - выполняет набор действий для каждого элемента в заданной последовательности. Цикл for обычно используется, когда заранее известно количество повторений или когда нужно перебрать элементы в списке, кортеже или другой последовательности.

2) Цикл while - выполняет набор действий до тех пор, пока условие истинно. Цикл while обычно используется, когда количество повторений неизвестно заранее или когда нужно повторять действия до выполнения определенного условия.

14. Назовите назначение и способы применения функции range.

Функция range в языке программирования Python используется для создания последовательности чисел. Она имеет следующий синтаксис:

`range(start, stop, step)`

1) start (необязательный) - начальное значение последовательности (по умолчанию 0).

2) stop (обязательный) - конечное значение последовательности (не включается в последовательность).

3) step (необязательный) - шаг, с которым генерируются числа (по умолчанию 1).

Функция range может использоваться для создания циклов, перебора значений, генерации списков и других задач, где требуется последовательность чисел.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

Для организации перебора значений от 15 до 0 с шагом 2 с помощью функции range, можно использовать следующий код:

```
for i in range(15, -1, -2):  
    print(i)
```

16. Могут ли быть циклы вложенными?

Да, циклы в языке программирования Python могут быть вложенными, то есть один цикл может находиться внутри другого цикла. Вложенные циклы позволяют выполнять повторяющиеся действия внутри других повторяющихся действий.

17. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл образуется, когда условие цикла всегда остается истинным, и цикл продолжает выполняться бесконечно. Например, если условие цикла всегда равно True или если условие никогда не изменяется, то цикл будет выполняться бесконечно.

Чтобы выйти из бесконечного цикла, можно использовать операторы break или условие, которое станет ложным в определенный момент времени. Оператор break позволяет прервать выполнение цикла и выйти из него досрочно, даже если условие цикла остается истинным.

18. Для чего нужен оператор break?

Оператор break используется в циклах для прерывания выполнения цикла и выхода из него досрочно. Когда оператор break достигается внутри

цикла, выполнение цикла немедленно прекращается, и управление передается к следующей инструкции после цикла.

Оператор `break` полезен, когда требуется прервать выполнение цикла, когда определенное условие выполняется, или когда достигнута определенная точка в программе, и дальнейшее выполнение цикла не требуется.

19. Где употребляется оператор `continue` и для чего он используется?

Оператор `continue` используется в циклах для пропуска текущей итерации цикла и перехода к следующей итерации. Когда оператор `continue` достигается внутри цикла, оставшаяся часть текущей итерации пропускается, и управление передается к следующей итерации цикла.

Оператор `continue` полезен, когда требуется пропустить выполнение определенных действий в цикле, но продолжить выполнение цикла с следующей итерации. Например, можно использовать оператор `continue` для пропуска выполнения некоторых действий в цикле, если определенное условие выполняется.

20. Для чего нужны стандартные потоки `stdout` и `stderr`?

Стандартные потоки `stdout` (стандартный вывод) и `stderr` (стандартный поток ошибок) являются каналами, используемыми для вывода информации из программы.

1) `stdout` - используется для вывода обычной информации или результатов работы программы. Этот поток обычно направляется в консоль или другое устройство вывода.

2) `stderr` - используется для вывода сообщений об ошибках или другой информации об ошибках, которые могут возникнуть в программе. Этот поток обычно также направляется в консоль или другое устройство вывода, но может быть перенаправлен в файл или другой поток.

21. Как в Python организовать вывод в стандартный поток `stderr`?

В Python для организации вывода в стандартный поток `stderr` можно использовать модуль `sys`. Вот пример кода:

```
import sys
```

```
sys.stderr.write("Это сообщение об ошибке\n")
```

В этом примере используется функция `write` из модуля `sys`, чтобы написать сообщение в стандартный поток ошибок `stderr`. Сообщение будет выведено в консоль или другое устройство вывода, на которое направлен поток ошибок.

22. Каково назначение функции `exit`?

Функция `exit` используется в Python для немедленного завершения программы. Когда функция `exit` вызывается, выполнение программы прекращается, и программа выходит из программы с указанным кодом завершения.

Назначение функции `exit` заключается в том, чтобы предоставить возможность явно завершить программу в определенных ситуациях. Например, если возникла критическая ошибка или требуется принудительно остановить выполнение программы, можно вызвать функцию `exit` с соответствующим кодом завершения.