Лабораторная работа №5

По дисциплине: Основы программной инженерии

Духно Михаил

ПИЖ-б-о-20-1

Программа task_1.py, код и результат работы

Задача: Дано число ( ). Определить полугодие, на которое приходится месяц с номером и количество дней в том месяце (год не високосный).

```python
if __name__ == '__main__':
    month = input('Enter number of month: ')
    month = int(month)

    if month > 6:
        print('Second part of year')
    else:
        print('First part of year')

    if month == 2:
        print('28 days in this month')
    elif month % 2 == 0:
        print('31 days in this month')
    else:
        print('30 days in this month')
```

Рисунок 5.1 – Код программы task_1.py

```
C:\Users\я\AppData\Local\Programs\P
Enter number of month: 10
First part of year
31 days in this month

Process finished with exit code 0
```

Рисунок 5.2 – Результат работы программы task_1.py

Программа task_2.py, код и результат работы

Задача: Решить неравенство , где - произвольное действительное число.

```python
import math


if __name__ == '__main__':
    a = int(input('Write a: '))
    x = 9999

    flag = False

    while not flag:
        try:
            flag = math.sqrt(a - x) > x - 2
        except:
            x -= 1
            continue

        if flag:
            break
        else:
            x -= 1

    print(f'x = {x}')
```

Рисунок 5.3 – Код программы task_2.py

```
C:\Users\я\AppData\Local\Programs\P
Write a: 5
x = 3

Process finished with exit code 0
```

Рисунок 5.4 – Результат работы программы task_2.py

Программа task_3.py, код и результат работы

Задача: Сумма цифр трехзначного числа кратна 7. Само число также делится на 7. Найти все такие числа.

```python
if __name__ == '__main__':
    for i in range(100, 999):
        number = str(i)
        if int(number) % 7 == 0 and (int(number[0]) + int(number[1]) + int(number[2])) % 7 == 0:
            print(i)
```

Рисунок 5.5 – Код программы task_3.py

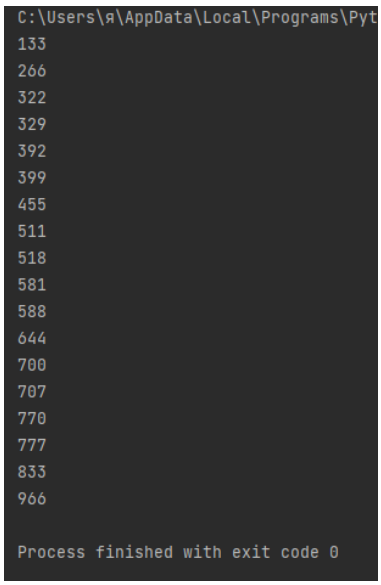

Рисунок 5.6 – Результат работы программы task_3.py

Программа hardLevel.py, код и результат работы

Интеграл вероятности:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\,dt = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)n!}.$$

Задача:

Рисунок 5.7 – Код программы hardLevel.py

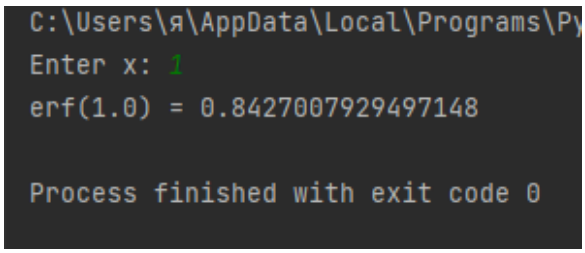

Рисунок 5.8 – Результат работы программы hardLevel.py

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

2. Что такое состояние действия и состояние деятельности?

Состояния деятельности и **действия Состояние деятельности** (activity state)
- **состояние** в графе **деятельности**, которое служит для представления процедурной последовательности **действий**, требующих определенного времени.

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

деятельности?

При построении диаграммы деятельности используются только нетриггерные переходы, т. е. такие, которые происходят сразу после завершения деятельности или выполнения соответствующего действия. Такой переход передает управление в последующее состояние сразу, как только закончится действие или *деятельность* в предыдущем состоянии. На диаграмме такой переход изображается сплошной линией со стрелкой.

Графически *ветвление* на диаграмме деятельности обозначается символом *решения* (*decision*), изображаемого в форме небольшого ромба, внутри которого нет никакого текста.

Для обозначений переходов используется стрелка указывающая на дальнейшее действие

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Разветвляющийся алгоритм – это **алгоритм**, в котором последовательность выполнения операций зависит от определенных условий. Если в **алгоритме** присутствует «действие1» и «действие2» (то есть ветвь 1 и ветвь 2), то это **разветвляющийся алгоритм** с полной альтернативой.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно.

Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.

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

Условный оператор – оператор который проверяет значение одного значение с другим значением.

Существуют операторы if else elif

В операторах условий можно составлять сложные условия

7. Какие операторы сравнения используются в Python?

Существуют операторы if else elif

8. Что называется простым условием? Приведите примеры.

if a == b:

9. Что такое составное условие? Приведите примеры.

if a == b and b not in list or a in list:

10. Какие логические операторы допускаются при составлении сложных условий?

Коньюнкция, дизьюнкция, инверсия

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

Может

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры – это **алгоритм**, в котором предусмотрено неоднократное выполнение одной и той же последовательности действий.

13. Типы циклов в языке Python.

С предусловием

С щётчиком

Перебор элементов коллекций

14. Назовите назначение и способы применения функции range .

Для определения промежутка чисел

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

for i in range(15, 0, 2):

16. Могул ли быть циклы вложенными?

Могут

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

While true

Выйти можно при помощи break

18. Для чего нужен оператор break ?

Для преждевременной остановки выполнения цикла

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

В циклах чтобы пропустить итерацию и приступить к следующей итерации

20. Для чего нужны стандартные потоки stdout и stderr?

stdin , **stdout и stderr** — три **потока** данных, созданные при запуске команды Linux.

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

```
import sys


class MyStdout(object):
    def __init__(self, old_stdout):
        self.true_sttdout = old_stdout

    def write(self, msg):
        self.true_sttdout.write(msg[::-1])

    def flush(self):
        self.true_sttdout.flush()
```

```
sys.stdout = MyStdout(sys.stdout)
print("it works :)")
```

22. Каково назначение функции exit ?

Функция `exit()` модуля `sys` - выход из Python.