Инвариантная самостоятельная работа №2

Тема: Понятие функции, объявление функций
Оборудование: VS code, python3

Задача 2.1-2.2

Постановка задачи:Разработать скрипт с функцией, которая строит таблицу истинности для логического выражения (по вариантам) для двух и трех аргументов (используются различные наборы значений аргументов). Разработать программу, которая выводит на экран с помощью ASCII-графики таблицу истинности на основе переданных ей на вход аргументов (логическое выражение, аргументы, результат вычисления выражения). Формирование отчета по выполнению задания и размещение его в портфолио, персональном репозитории.

Код программы:

```
"""
Царулкова Анастасия Витальевна
2 группа 3 подгруппа

Copyright: 10.2019

Задание 2.1 + 2.2 ИСР
Скрипт с функцией, которая строит таблицу истинности для логического выражения.

"""

def logic_values(l, *args):
    return int(l(*args))

def printHeader(*args):
    result = ''
    lst_of_len = []
    for el in args:
        lst_of_len.append(len(str(el))+2)
    lst_of_result = []
    for el in range(len(lst_of_len)):
        lst_of_result.append(str(args[el]).center(lst_of_len[el]))

    for el in range(len(lst_of_result)):
        result += '|'+lst_of_result[el]
    result += '|'
```

```
    row_length = len(result)
    print('-'*row_length)
    print(result)
    return lst_of_len

def printRow(lst, *args):
    result = ''
    lst_of_result = []
    for el in range(len(lst)):
        lst_of_result.append(str(args[el]).center(lst[el]))

    for el in range(len(lst_of_result)):
        result += '|'+lst_of_result[el]
    result += '|'
    row_length = len(result)
    print('-'*row_length)
    print(result)

def main():
    A = [0, 0, 0, 0, 1, 1, 1, 1]
    B = [0, 0, 1, 1, 0, 0, 1, 1]
    C = [0, 1, 0, 1, 0, 1, 0, 1]

    temp_list = printHeader('A', 'B', 'C', '¬C', 'A ^ B', 'B ^ C',
                        '¬C → A', 'A ^ B ↔ B ^ C', '(A ^ B ↔ B ^ C) ˇ (¬C -> A)')

    for el in range(len(A)):
        ans1 = logic_values(lambda c: not c, C[el])
        ans2 = logic_values(lambda a, b: a and b, A[el], B[el])
        ans3 = logic_values(lambda b, c: b and c, B[el], C[el])
        ans4 = logic_values(lambda c, a: c or a, ans1, A[el])
        ans5 = logic_values(lambda a, b: a == b, ans2, ans3)
        ans6 = logic_values(lambda a, b: a or b, ans5, ans4)
        printRow(temp_list, A[el], B[el], C[el],
                ans1, ans2, ans3, ans4, ans5, ans6)
```

Результат работы программы:

A	B	C	¬C	A ˄ B	B ˄ C	¬C → A	A ˄ B ↔ B ˄ C	(A ˄ B ↔ B ˄ C) ˅ (¬C -> A)
0	0	0	1	0	0	1	1	1
0	0	1	0	0	0	0	1	1
0	1	0	1	0	0	1	1	1
0	1	1	0	0	1	0	0	0
1	0	0	1	0	0	1	1	1
1	0	1	0	0	0	1	1	1
1	1	0	1	1	0	1	0	1
1	1	1	0	1	1	1	1	1

Задача 2.3

Постановка задачи: Разработать скрипт с функцией, которая для ряда Фибоначчи, где количество элементов, n = 22, возвращает подмножество значений или единственное значение (по вариантам). Для нахождения элемента требуется использовать слайсы. Формирование отчета по выполнению задания и размещение его в портфолио, персональном репозитории. (Сделаны все варианты)

Код программы:

```
def fibon_1(lst):
    sub_List = lst[2::2]
    result = sum(sub_List)
    print(result)

def fibon_2(lst):
    sub_List = lst[5::2]
    result = sum(sub_List)
    print(result)

def fibon_3(lst):
    med = round(len(lst)/2)
    sub_List = lst[:med:2]
    result = sum(sub_List)
    print(result)

def fibon_4(lst):
    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med+1::2]
```

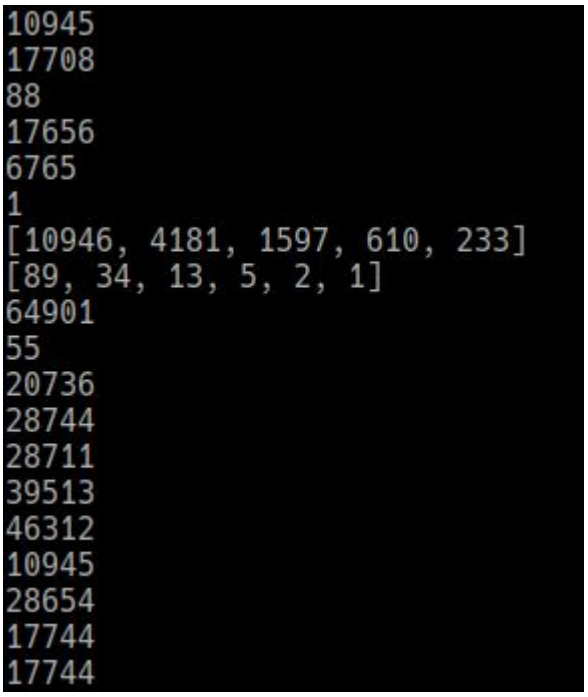
```
    else:
        sub_List = lst[med::2]
    result = sum(sub_List)
    print(result)

def fibon_5(lst):
    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med::2]
    else:
        sub_List = lst[med+1::2]
    result = max(sub_List)
    print(result)


def fibon_6(lst):
    med = round(len(lst)/2)
    sub_List = lst[1:med:2]
    result = min(sub_List)
    print(result)

def fibon_7(lst):
    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[len(lst)-1:med:-2]
    else:
        sub_List = lst[len(lst)-1:med+1:-2]

    print(sub_List)

def fibon_8(lst):
    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med+1:0:-2]
    else:
        sub_List = lst[med:0:-2]

    print(sub_List)

def fibon_9(lst):

    med = round(len(lst)/2)

    sub_List = lst[med:lst[len(lst)-1]]

    result = str(sub_List[len(sub_List)-1])
```

```
    result1 = result[::-1]

    print(result1)

def fibon_10(lst):

    med = round(len(lst)/2)

    sub_List = lst[:med:]
    result = sub_List[len(sub_List)-1]
    print(result)

def fibon_11(lst):

    med = round(len(lst)/2)

    sub_List = lst[med-3:med+3]

    even_List = sub_List[::2]

    result = even_List[len(even_List)-1]**2
    print(result)

def fibon_12(lst):
    med = round(len(lst)/2)
    sub_List = lst[:med:2]
    result1 = sum(sub_List)

    result = result1 + sum(lst)
    print(result)

def fibon_13(lst):
    med = round(len(lst)/2)
    sub_List = lst[1:med:2]
    result1 = sum(sub_List)
    result = result1 + sum(lst)
    print(result)

def fibon_14(lst):
    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med::2]
    else:
        sub_List = lst[med+1::2]
    result = sum(sub_List) + sum(lst)
    print(result)
```

```
def fibon_15(lst):
    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med+1::2]
    else:
        sub_List = lst[med::2]
    result1 = sum(sub_List)

    result = sum(lst) + result1
    print(result)

def fibon_16(lst):
    sub_List = lst[2::2]
    result1 = sum(sub_List)

    result = min(lst) + result1
    print(result)


def fibon_17(lst):
    sub_List = lst[5::2]
    result1 = sum(sub_List)

    result = max(lst) + result1
    print(result)


def fibon_18(lst):
    med = round(len(lst)/2)
    sub_List = lst[:med:2]
    result1 = sum(sub_List)

    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med+1::2]
    else:
        sub_List = lst[med::2]
    result2= sum(sub_List)

    result = result1 + result2
    print(result)

def fibon_19(lst):
    med = round(len(lst)/2)
    sub_List = lst[:med:2]
```

```
    result1 = sum(sub_List)

    med = round(len(lst)/2)
    if med % 2 == 0:
        sub_List = lst[med+1::2]
    else:
        sub_List = lst[med::2]
    result2= sum(sub_List)

    result = result1 + result2
    print(result)

def main():
    lst = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610,
    987, 1597, 2584, 4181, 6765, 10946]

    fibon_1(lst)
    fibon_2(lst)
    fibon_3(lst)
    fibon_4(lst)
    fibon_5(lst)
    fibon_6(lst)
    fibon_7(lst)
    fibon_8(lst)
    fibon_9(lst)
    fibon_10(lst)
    fibon_11(lst)
    fibon_12(lst)
    fibon_13(lst)
    fibon_14(lst)
    fibon_15(lst)
    fibon_16(lst)
    fibon_17(lst)
    fibon_18(lst)
    fibon_19(lst)

main()
```

Результат работы программы:



Задача 1.4

Постановка задачи: Напишите программу с функцией, в которой будет реализовано решение физической задачи (по вариантам). Например: ящик, имеющий форму куба с ребром a см без одной грани, нужно покрасить со всех сторон снаружи. Найдите площадь поверхности, которую необходимо покрасить. Ответ дайте в квадратных сантиметрах. Решение задачи оформите в виде функции square(a), которая возвращает значение s. Например, при значении a = 30, square(30) вернет s = 4500. Формирование отчета по выполнению задания и размещение его в портфолио, персональном репозитории.

Код программы:

```
"""
Царулкова Анастасия Витальевна
2 группа 3 подгруппа

Вариант 11
Основанием прямой треугольной призмы служит прямоугольный треугольник с катетами a и b.
Площадь ее поверхности равна s. Найдите высоту призмы.
Решение оформите в виде функции height(a,b,s), которая возвращает значение переменной h.
Например, при a=6; b=8; s=288 функция height(6,8,288) выдает h=10.
```

```
"""
import math

def height(a, b, s):
    return (s-a*b)/(a+b+math.sqrt(a*a+b*b))

def main():
    a = int(input("Введите сторону треугольника a = "))
    b = int(input("Введите сторону треугольника b = "))
    s = int(input("Введите площадь призмы s = "))
    print(height(a, b, s))

main()
```

Результат работы программы