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

Cкрипт 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) ˅ (¬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])

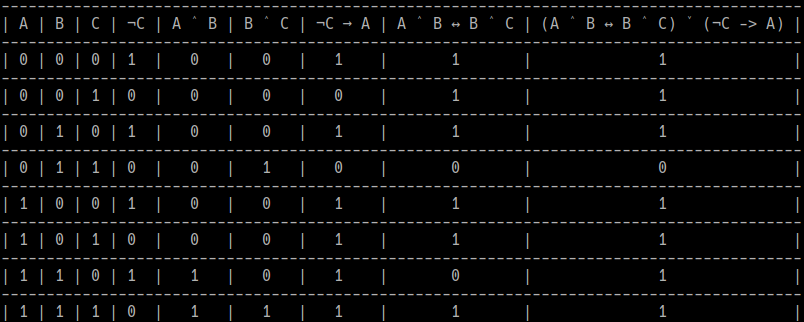
ans4 = logic\_values(lambda c, a: c or a, ans1, A[el])

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:**

**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)

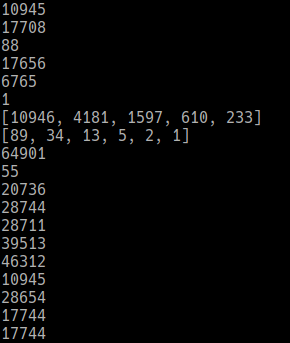
fibon\_17(lst)

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**

