Федеральное государственное автономное образовательное учреждение высшего образования

«ОМСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика и фундаментальная информатика»

Практическое занятие №9

по дисциплине «Практикум по программированию»

на тему: «Функции в Python: лямбда-функции. Библиотеки в Python»

Вариант №14

Выполнил

Студент гр. **ФИТ-212** **Курпенов К.И.**

группа Фамилия И,О. студента подпись

Принял:

Преподаватель **Моисеева Н.А.**

Фамилия И.О, преподавателя дата, подпись

Омск 2022

ЗАДАНИЕ 1.

Задача 1.1.

Условие:

Напишите программу для создания лямбда-функции, которая добавляет 15 к заданному числу, переданному в качестве аргумента, а также создайте лямбда-функцию, которая умножает аргумент x на аргумент y и печатает результат.

Решение:

```
addtion = lambda x: x + 15

multiplication = lambda x, y: print(x * y)

if __name__ == "__main__":
    print(addtion(10))
    multiplication(6, 8)
```

Задание 1.2.

Условие:

Напишите программу для сортировки каждого подсписка строк в заданном списке списков с помощью лямбда-выражения.

Решение:

```
sort = lambda array: [list(sorted(sublist)) for sublist in array]

if __name__ == "__main__":
    print(sort([
        ["green", "orange"],
        ["black", "white"],
        ["white", "black", "orange"]
    ]))
```

Задание 1.3.

Условие:

Напишите программу для сортировки списка кортежей с помощью лямбда-функции.

Решение:

```python
sort = lambda array: [sublist for sublist in sorted(array, key=lambda x: x[1])]

if __name__ == "__main__":
    print(sort([
        ("English", 88),
        ("Social", 82),
        ("Science", 90),
        ("Math", 97)
    ]))
```

Задание 1.4.

Условие:

Напишите программу для сортировки списка словарей с помощью Lambda.

Решение:

```python
sort = lambda array: [element for element in
        sorted(array, key=lambda x: x["model"], reverse=True)]

if __name__ == "__main__":
    print(sort([
        {"make": "Nokia", "model": 216, "color": "black"},
        {"make": "Mi Max", "model": 2, "color": "gold"},
        {"mako": "Samsung", "model": 7, "color": "blue"}
    ]))
```

Задание 1.5.

Условие:

Напишите программу, чтобы найти вторую самую низкую оценку любого учащегося по заданным именам и оценкам каждого учащегося, используя списки и лямбда. Введите количество учеников, имена и оценки каждого ученика.

Решение:

```python
if __name__ == "__main__":
    count = int(input("[>] Enter count of students: "))

    students = dict()

    for i in range(count):
        name = input("[>] Enter name: ")
        estimation = int(input("[>] Enter estimation: "))
        students[name] = estimation

    print(f"[+] Second minimal: {list(sorted(students.keys()))[1]}")
```

Задание 1.6.

Условие:

Напишите программу, чтобы найти числа заданной строки и сохранить их в списке, отображать числа, которые больше, чем длина списка, в сортированном виде. Используйте лямбда-функцию для решения проблемы.

Решение:

```python
def save_numbers(enter: list) -> list:
    numbers = []

    for word in enter:
        if word.isdigit() and int(word) :
            numbers.append(word)

    return numbers


if __name__ == "__main__":
    print(save_numbers("sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5".split()))
```

Задание 1.7.

Условие:

Напишите программу на Python для сортировки заданного списка списков по длине и значению с использованием лямбда.

Решение:

```
1 sort = lambda array: [sublist for sublist in sorted(array, key=lambda x: (len(x),
   x[0]))]
2
3 if __name__ == "__main__":
4     print(sort([[2], [0], [1, 3], [0, 7], [9, 11], [13, 15, 17]]))
5
```

Задание 1.8.

Условие:

Напишите программу, чтобы найти максимальное значение в заданном разнородном списке, используя лямбда.

Решение:

```
1 finder = lambda array: max(list(sorted([i for i in array if str(i).isdigit()])))
2
3 if __name__ == "__main__":
4     print(finder(["Python", 3, 2, 4, 5, "version"]))
5
```

Задание 1.9.

Условие:

Напишите программу для подсчета чисел с плавающей запятой в заданном смешанном списке с использованием лямбда.

Решение:

```
1 counter = lambda array: len([i for i in array if type(i) == float])
2
3 if __name__ == "__main__":
4     print(counter([1, "abcd", 3.12, 1.2, 4, "xyz", 5, "pqr", 7, -5, -12.22]))
5
```

ЗАДАНИЕ 2.

Задание 2.1.

Условие:

У каждого жителя планетной системы α−ω−Fomalgaut−4α−ω−Fomalgaut−4 (*Альфа-Омега-Фомальгаут-4*) обязательно должно быть уникальное имя, построенное по определенным правилам. Поскольку их символы отсутствуют на нашей клавиатуре, заменим их нашими условно.

Правила такие:

– имя должно состоять из двух частей, то есть содержать один пробел; первая часть не может быть короче 2 символов, вторая может состоять из одного символа;

– в первой части обязательно должна быть цифра, но она не должна стоять на первом месте; остальные символы могут быть буквами латинского алфавита в любом регистре или цифрами;

– вторая часть должна начинаться с буквы из первой половины латинского алфавита в верхнем регистре; остальные символы могут быть любыми буквами латинского алфавита в нижнем регистре.

Напишите функцию **name()**, подбирающую имя для зеленого человечка по этим правилам по заданной длине (не меньше 4 символов, включая пробел).

Решение:

```python
from random import choices
from random import randint

from string import digits
from string import ascii_letters
from string import ascii_lowercase
from string import ascii_uppercase


def generate_name(lenght: int) -> str:
    lenght_part_1 = randint(2, lenght - 2)
    lenght_part_2 = lenght - lenght_part_1

    part_1 = ""
    part_2 = ""
    number_index = randint(1, lenght_part_1)

    for i in range(lenght_part_1):
        if not i:
            part_1 += "".join(choices(ascii_letters))
        elif i == number_index:
            part_1 += "".join(choices(digits))
        else:
            part_1 += "".join(choices(ascii_letters + digits))

    for i in range(lenght_part_2):
        if not i:
            part_2 += "".join(choices(
                ascii_uppercase[:len(ascii_uppercase) // 2]))
        else:
            part_2 += "".join(choices(ascii_lowercase))

    return f"{part_1} {part_2}"


if __name__ == "__main__":
    print(generate_name(12))
```

Задание 2.2.

Условие:

Если на картинке самое главное находится в центре, а остальное пусто или не интересно, то можно вырезать центральную часть и заключить в рамочку.

Напишите функцию frame(), которая принимает имя файла и ширину рамки, вырезает центральную часть изображения (треть размера по ширине и по высоте, используйте целочисленное деление) и добавляет рамку переданной ширины по контуру, цвет рамки – средний цвет вырезанной центральной части. Затем сохраняет его в файл done.png.

Для определения среднего цвета нужно найти сумму красной составляющей всех пикселей и поделить нацело на количество пикселей, затем то же сделать для остальных составляющих.

Решение:

```
  1 from PIL import Image
  2
  3
  4 def frame(path: str, width: int) -> None:
  5     image = Image.open(path)
  6     x, y = image.size
  7     pixels = image.load()
  8
  9     rgb = [0, 0, 0]
 10
 11     for i in range(x):
 12         for j in range(y):
>> 13             r, g, b = pixels[i, j]
 14             rgb[0] += r
 15             rgb[1] += g
 16             rgb[2] += b
 17
 18     rgb[0] //= x * y
 19     rgb[1] //= x * y
 20     rgb[2] //= x * y
 21
 22     croped = image.crop((x // 3, y // 3, x - x // 3, y - y // 3))
 23
 24     done = Image.new(
 25             "RGB",
 26             (x // 3 + (2 * width), y // 3 + (2 * width)),
 27             tuple(rgb))
 28     done.paste(croped, (20, 20))
 29     done.save("done.png")
 30
 31
 32 if __name__ == "__main__":
 33     frame("bug.png", 20)
 34
```

Задание 2.3.

Условие:

Нарисуйте паровозик по размерам и цветам, указанным на рисунке. Размер рисунка 280x200. Для этого напишите функцию train(), принимающую имя файла для сохранения.

Решение:

```python
from PIL import Image, ImageDraw


def train():
    pass


if __name__ == "__main__":
    train()
```

Задание 2.4.

Условие:

Напишите программу, которая найдет в предложении существительное и число (записано цифрами) и согласует их между собой. Для некоторого упрощения из предложения убраны знаки препинания и все слова записаны с маленькой буквы.

Решение:

```
 1 import pymorphy2
 2
 3
 4 def get_nouns(lines: list) -> list:
 5     morph = pymorphy2.MorphAnalyzer()
 6
 7     numbers = []
 8     nouns = []
 9
10     for string in lines:
11         words = string.split()
12         for word in words:
13             parser = morph.parse(word)[0]
14             if word.isdigit():
15                 numbers.append(int(word))
>> 16             elif "NOUN" in parser.tag:
>> 17                 nouns.append(morph.parse(word)[0].normal_form)
18
19     result = []
20
21     for number, noun in zip(numbers, nouns):
22         comment = morph.parse(noun)[0]
>> 23         result.append(f"{number} {comment.make_agree_with_number(number).word}"
   )
24
25     return result
26
27
28 if __name__ == "__main__":
29     print(get_nouns(["варкалось шорька пырялись 1"]))
30
```

Задание 2.5.

Условие:

Первый же встреченный И.Тихим тарраканин любезно согласился поделиться каплей крови, или что там у него ее заменяет, для всестороннего анализа и сравнения с человеческой. Напишите программу, создающую бланк для заполнения анализа данными в виде таблицы. Вверху документа заголовок нулевого уровня, выравнивание по центру, остальное форматирование по умолчанию: Blood test

Затем таблица, верхняя строка – заголовки, полужирное начертание, выравнивание по центру ячейки: indicator, norm, value.

Сохраните документ в файл analysis.docx.

Решение:

```python
from docx import Document

document = Document()

document.add_heading("Blood test", 0)

records = [
        "WBC",
        "RBC",
        "HGB",
        "HTC",
        "MCV",
        "MCH",
        "MCHC",
        "RDW",
        "PLT",
        "MPV",
        "PTC",
        "NEU",
        "LYMP",
        "MONO",
        "EFO",
        "Baso"
]

lenght = len(records)

table = document.add_table(rows=1, cols=3)

header = table.rows[0].cells
header[0].text = "indicator"
header[1].text = "noem"
header[2].text = "value"

for i in range(lenght):
    rows = table.add_row().cells
    rows[0].text = records[i]

document.save("analysis.docx")
```