Python Speedrun Курс (3 Часа)
Практические Задания

https://www.youtube.com/watch?v=KpuueG1431Y

Оглавление

Список заданий по темам из курса
https://youtu.be/L2Ibq1pbvdU
с готовыми решениями.
Для каждой темы есть 1 или больше практических заданий, каждое из которых я советую попробовать выполнить самостоятельно после прохождения соответствующей темы

1. Переменные

```
##Создайте переменные для хранения вашего имени, возраста и любимого цвета. Выведите их на экран.


### Решение:

name = "Your name"
age = 42
favorite_colour = "Black"
print(name)
print(age)
print(favorite_colour)

## Создайте числовые переменные a и b и сделайте переменную c равной их разнице. Выведите c на экран

### Решение

a = 25
b = 12
c = a - b
print(c)
```

2. Типы данных

```
## Создайте список, содержащий разные типы данных (строку, число, список, кортеж и т.д.) и определите их типы с помощью функции type(). Выведите эти типы на экран

### Решение

list_with_different_types = ["I am string", 25, [1, 2, 3], ("John", "Bob", "Alice", "Mark"), True, 2.28]
# без использования циклов
print(type(list_with_different_types[0]))
print(type(list_with_different_types[1]))
print(type(list_with_different_types[2]))
print(type(list_with_different_types[3]))
print(type(list_with_different_types[4]))
print(type(list_with_different_types[5]))
print() # пустая строка
# с циклом
for element in list_with_different_types:
  print(type(element))
```

3. Условия

```
## Напишите программу, которая просит пользователя ввести число и определяет, четное ли это число или нет. Выведите результат

### Решение

number = int(input("Введите число: "))
if number % 2 == 0:
  print("Число четное")
else:
  print("Число нечетное")
```

4. Циклы

```
## Напишите программу, которая выведет числа от 1 до 10, и если число равно 7, то дополнительно выведет надпись - "7 - счастливое число". (Задание со звездочкой - попробуйте составить эту программу с использованием continue)

### Решение

for i in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):
  if i != 7:
    print(i)
    continue
  print(i, "7 - счастливое число")


## Создайте пустой список. Пока длина этого списка не станет равна 10 - наполняйте его числами начиная с нуля и увеличивая значение этого числа каждый раз на три. В конце выведите получившийся список

### Решение

some_list = []
to_append = 0
while len(some_list) < 10:
  some_list.append(to_append)
  to_append += 3
print(some_list)
```

5. Рэнжи и слайсы

```
## Создайте с помощью рэнжа кортеж чисел от 20 до 50 включительно
и сделайте срез этого кортежа, который будет содержать только
четные числа этого кортежа в обратном порядке. Выведите
получившийся результат на экран

### Решение

my_tuple = tuple(range(20, 51))
my_slice = my_tuple[::-2]
print(my_slice)
```

6. Функции

```
## Напишите функцию, которая принимает два аргумента - строку и
число. Эта функция должна вывести выбранную строку в консоль
указанное число раз.

### Решение

def print_word_n_times(word, times):
  for time in range(times):
    print(word)

print_word_n_times("Yo", 10)

## Напишите функцию, которая принимает любое кол-во числовых
аргументов и возвращает их произведение

### Решение

def multiply_numbers(*args):
  result = 1
  for arg in args:
    result *= arg
  return result

multiply_numbers(1, 2, 3, 4, 15)
```

#7. Встроенные функции

```
## Сделайте функцию, которая будет возвращать длину переданного ей
как аргумент имени

### Решение

def get_name_length(name):
  return len(name)

get_name_length("Denis")
```

```
## *Сделайте функцию, которая принимает ваш номер телефона в виде строки (вместе с + если он есть). Превращает этот номер в отсортированный список чисел и возвращает его. Так же в консоль выводится максимальная цифра в номере и минимальная, а также количество цифр, которые встречаются в вашем номере более одного раза.

### Решение

def get_sorted_phone_number(number: str):
  number_as_list = list(number)
  only_digits = []
  for element in number_as_list:
    if element in "0123456789":
      only_digits.append(int(element))

  print(f"Максимальная цифра: {max(only_digits)}")
  print(f"Минимальная цифра: {min(only_digits)}")
  print(f"Кол-во дубликатов: {len(only_digits) -
len(set(only_digits))}")

  return sorted(only_digits)

print(get_sorted_phone_number("+5 492 255 33 84"))
print(get_sorted_phone_number("+9-666-123-45-67"))
```

8. Пространства имен

```
## Изучите локальные и глобальные переменные в функции, попробуйте изменить глобальную переменную из функции. (Погуглите ключевые слова global  и nonlocal)

# 1) измените программу так, чтобы вывод был
# local x
# enclosing x
# local x
# сами переменные менять нельзя

# 2) измените программу так, чтобы вывод был
# local x
# local x
# global x
# сами переменные менять нельзя


x = "global x"

def my_func_1():
  x = "enclosing x"
  def my_func_2():
    x = "local x"
    print(x)
  my_func_2()
```

```
  print(x)

my_func_1()
print(x)

### Решение 1

x = "global x"

def my_func_1():
  x = "enclosing x"
  def my_func_2():
    global x
    x = "local x"
    print(x)
  my_func_2()
  print(x)

my_func_1()
print(x)

### Решение 2

x = "global x"

def my_func_1():
  x = "enclosing x"
  def my_func_2():
    nonlocal x
    x = "local x"
    print(x)
  my_func_2()
  print(x)

my_func_1()
print(x)
```

9. Компрехеншены

```
## Скопируйте любой абзац текста из интернета и сохраните его как
многострочную строку. Создайте с помощью лист-компрехеншена
модифицированный текст, каждое слово в котором заканчивалось бы
какой-то дополнительной частицей речи (например ругательством:)

### Решение

long_string = """Разведение гусей в России издавна считалось
выгодным занятием, благодаря использованию в питании гусей
малоценных кормов, получению высококачественного мяса и жира, а
так же ценного перо-пухового сырья для легкой промышленности.
В сравнении с другими домашними птицами, гуси неприхотливы,
способны в большом количестве поедать и переваривать зеленую
траву, различные корнеплоды и даже сено. Это дает возможность
```

```
разводить гусей в тех местах, где имеются малоценные пастбища и
неудобья (склоны, овраги).
Гуси - быстрорастущая птица, живая масса гусят суточных до
возраста 70-90 дней вырастает в 50 раз!
Масса 3-месячного подращенного гусенка бывает до 5кг. Удивительна
и конверсия корма у гусят, съеденные 3кг комбикорма, дают 1кг
прибавки веса!"""

modified_text = "".join([word + " йопта " for word in
long_string.split()])
print(modified_text)
```

10. Исключения

```
## Напишите функцию, которая вызывает input с просьбой ввести ваш
возраст. Если полученный ответ невозможно конвертировать в число,
то функция должна вывести ошибку и предлагать ввести возраст пока
юзер не введет валидное число

### Решение

def unbreakable_age_input():
  age = None
  while True:
    try:
      age = int(input("Введите ваш возраст"))
      break
    except ValueError:
      continue
  print(f"Ваш возраст: {age}")

unbreakable_age_input()

# 11. Генераторы, итераторы

## Создайте генератор, в котором будет имплементирован словарь, в
котором ключ - это название месяца, а значение - кол-во дней в
этом месяце. Этот генератор должен возвращать сообщение по типу "В
месяце Январь 31 день" или в "В месяце Апрель 30 дней". Выведите с
помощью цикла for все доступные значения из генератора.

### Решение

def monthly_message_generator():
  months_to_days_count_map = {
      "Январь": 31,
      "Февраль": 28,
      "Март": 31,
      "Апрель": 30,
      "Май": 31,
      "Июнь": 30,
      "Июль": 31,
      "Август": 31,
```

```
      "Сентябрь": 30,
      "Октябрь": 31,
      "Ноябрь": 30,
      "Декабрь": 31,
  }
  for month, days_count in months_to_days_count_map.items():
    yield f"В месяце {month} {days_count} {'дней' if days_count in
(28, 30) else 'день'}"

for message in monthly_message_generator():
  print(message)
```

12. ООП

```
## Создайте класс Car с атрибутами конкретной машины: марка, цвет
и скорость. Добавьте методы для увеличения и уменьшения скорости

### Решение

class Car:
  def __init__(self, brand, color, speed=0):
    self.brand = brand
    self.color = color
    self.speed = speed

  def gas(self):
    self.speed += 10
    return self

  def stop(self):
    self.speed = 0
    return self

my_car = Car("Volga", "Black")
my_car.gas().gas().gas()
print(my_car.speed)
my_car.stop()
print(my_car.speed)
```

Наследование.

Создайте класс ElectricCar, который наследует класс Car и добавляет атрибут заряда батареи.

Решение

```
# будет работать, если вы выполнили ячейку с кодом из предыдущего упражнения. Другими словами, класс Car уже должен быть объявлен

class ElectricCar(Car):
  car_type = "electric"


tesla = ElectricCar("Tesla", "Red")
tesla.gas()
print(tesla.car_type)
print(tesla.speed)
```

Инкапсуляция.

Сделайте скорость приватным атрибутом и модифицируйте методы газ и стоп. Изменяться скорость должна только при помощи этих методов. Скорость не должна превышать 100 км/час. Добавьте метод get_speed, который возвращает значение скорости в км/ч

Решение

```
class Car:
  def __init__(self, brand, color):
    self.brand = brand
    self.color = color
    self.__speed = 0

  def gas(self):
    if self.__speed < 100:
      self.__speed += 10
    else:
      print("Больше не разогнаться")
    return self

  def stop(self):
    self.__speed = 0
    return self

  def get_speed(self):
    return f"{self.__speed} км/ч"

my_car = Car("Volga", "Black")
for i in range(12):
  my_car.gas()
  print(my_car.get_speed())
my_car.stop()
print(my_car.get_speed())
```

Полиморфизм.

Создайте несколько классов животных с методом speak(), который возвращает разные звуки для разных животных. Создайте коллекцию из экземпляров этих классов и в цикле вызовите у каждого метод speak.

Решение

```
class Dog:
  def speak(self):
    return "Bark!"

class Cat:
  def speak(self):
    return "Meow"

class Cow:
  def speak(self):
    return "Mooo"

for animal in (Dog(), Cat(), Cow()):
  print(animal.speak())
```

13. Декораторы

Создайте декоратор timer, который засекает время выполнения декорируемой функции и выводит его на экран.

Решение

```
from os import times_result
import time

def timer(func):
  def wrapper(*args, **kwargs):
    time_start = time.perf_counter()
    result = func(*args, **kwargs)
    time_delta = time.perf_counter() - time_start
    print(f"Time delta: {round(time_delta, 5)} sec")
    return result
  return wrapper


@timer
def long_process():
  time.sleep(3)

long_process()
```

14. Контекстные менеджеры

```
## Создайте контекстный менеджер ExecutionTimer, который будет измерять время, проведенное внутри контекста, и выводить это время по завершении блока.

Пример работы
```
from time import sleep

with ExecutionTimer():
    sleep(2) # имитация какой-либо долгой операции

```Код выполнялся 2.00 секунды```

Решение

import time

class ExecutionTimer:
  def __enter__(self):
    self.time_start = time.perf_counter()

  def __exit__(self, a, b, c):
    print(f"Код выполнялся {round(time.perf_counter() - self.time_start, 2)} секунд/ы")
    return True

with ExecutionTimer():
  time.sleep(2)
```

15. Рекурсия

```
## Напишите рекурсивную функцию для вычисления факториала числа

### Решение

def get_factorial(n):
  if n in (0, 1):
    return n
  return n*get_factorial(n-1)

get_factorial(10)

# 16. Lambda функции

## Используйте лямбда-выражение для вычисления объема куба со стороной длины a

### Решение

get_cube_volume = lambda a: a**3
print(get_cube_volume(10))
```

17. ФП

```
## Создайте последовательность чисел от -10 до 10 включительно. С помощью функции map и вспомогательного callable сделайте эту последовательность чисел другой последовательностью, каждое число которой на 10 меньше предыдущего (если оно было отрицательное) и на 10 больше предыдущего (если он положительное).
В результате вы должны получить
[-20, -19, ... - 11, 0 , 11, 12 ... 19, 20]

### Решение

original_sequence = range(-10, 11)
def tricky_modifier(number):
  """
  return number + 10 if number is positive
  return number - 10 if number is negative
  else return 0
  """
  if number > 0:
    return number + 10
  elif number < 0:
    return number - 10
  return 0

print(list(map(tricky_modifier, original_sequence)))
```