Урок 1. Проектная деятельность. Повторение. Типы данных.

ФИО_____

Тест:

1. *Что выведет данный код?*

```
x = 0
if x or 5 and not 0:
    print('ok')
else:
    print('no')
```

_____

2. *Что вернёт* `print(bool(1 + 0))`*?*
3. *Что делает блок else у цикла for?*
   A. Всегда выполняется после цикла
   B. Выполняется, если цикл завершился **без** break
   C. Не выполняется
   D. Выполняется, если был continue
4. *Что такое асимптотика задач?*

---

5. *Что выведет данная программа?*

```
for x in [1, 2, 3]:
    if x % 2 == 0:
        break
else:
    print('done')
```

_____

6. *Чему будет равна сумма?*

```
s = sum(i for i in range(1, 6) if i % 2 == 0)
print(s)
```

---

7. *Укажите различия между break и continue:*

Теория:

Типы данных:
Числовые типы данных: int(), float()
Логические типы данных: bool()
Коллекции: list() tuple(), str(), set(), frozenset(), dict()

Обобщение свойств встроенных коллекций в сводной таблице:

Тип коллекции	Изменяемость	Индексированность	Уникальность	Как создаём
Список (list)	+	+	-	*[]* *list()*
Кортеж (tuple)	-	+	-	*() ,* *tuple()*
Строка (string)	-	+	-	*''* *""*
Множество (set)	+	-	+	*{elm1, elm2}* *set()*
Неизменное множество (frozenset)	-	-	+	*frozenset()*
Словарь (dict)	+ элементы - ключи + значения	-	+ элементы + ключи - значения	*{}* *{key: value,}* *dict()*

Условные конструкции:
Истинность/ложность ложными считаются `0`, `0.0`, `''`, `[]`, `{}`, `set()`, `None`, `False`. Всё остальное — истинно.
Сравнения и цепочки: 0 < x <= 10 # эквивалентно (0 < x) and (x <= 10)
Логические операторы: `and`, `or`, `not` возвращают **значения**, а не только `True/False`.
`name = user_name or 'Anonymous'` # если user_name пустой → 'Anonymous'

Пример конструкции:

```
if условие_1:
    ...
elif условие_2:
    ...
else:
    ...
# Тернарное выражение
res = A if условие else B
```

Циклы:

```
for i in range(n):
    ...
for idx, val in enumerate(arr):
    ...
for a, b in zip(A, B):
    ...
while условие:
    ...
# Управление потоком
for x in data:
    if условие_1:
        break      # прерывает весь цикл
    if условие_2:
        continue   # пропускает итерацию
else:
    # выполнится, если цикл завершился без break
    print('не было bad-элементов')
```

enumerate() – встроенная функция в Python, которая позволяет перебирать элементы списка вместе с их индексами.
zip() – встроенная функция в Python, которая объединяет элементы нескольких итерируемых объектов (списков, кортежей, строк) в пары.

Компрехеншены:

```
squares = [x*x for x in arr if x % 2 == 0]
unique = {w.lower() for w in words}
index = {v: i for i, v in enumerate(arr)}
```

Компрехеншены – это конструкции, которые позволяют создавать новые структуры данных на основе существующих итерируемых объектов**.** Ранее подобное мы называли генераторами и рассматривали частный случай такой конструкции: списковые включения.

Задачи:

1. **Сумма нечётных**: по заданному n вывести сумму всех нечётных от 1 до n включительно. Решить двумя способами: цикл for и формула.
2. **Фильтр по условию**: дан список чисел a. Выведите новый список из элементов, чьи квадраты меньше 100. (Решение через цикл и через списковое включение.)
3. **Мини-валидатор пароля**: строка s. Если длина >= 8 и есть цифра — «OK», иначе «Weak».
4. **Без подряд**: удалить подряд идущие дубликаты из списка, сохранив первый из блока.
   Пример: [1,1,2,2,2,3,1] → [1,2,3,1].
5. **Длина максимальной серии**: для списка целых найти максимальную длину серии одинаковых элементов.

6. **Первое простое в отрезке** [L, R]: вывести первое простое; если не найдено — «NONE». Использовать for-else.
7. **K-разность**: по массиву a и числу k посчитать количество пар (i < j) с |a[i]-a[j]| = k. Решить за O(n)по времени и O(n) по памяти (через множество/словарь).
8. **Слияние двух отсортированных**: дано два отсортированных списка. Слейте их в один отсортированный **без** sort (два указателя).
9. **Почти возрастающая**: дана последовательность a. Определите, можно ли удалить **не более одного** элемента так, чтобы последовательность стала строго возрастающей. Верните True/False.
10. **Чередование знаков**: по списку целых найдите длину **максимального** подотрезка, где знаки строго чередуются (+,-,+,-,... или -,+,-,+,...). Нули считаются «разрывом» (обнуляют текущую длину)