Лабораторна робота 3

Тема: Розгалуження, цикли та перебір даних.

Мета: Навчитися поєднувати умовні оператори з циклами, обчислювати поточні підсумки і аналізувати послідовність слів.

Зміст роботи

Завдання 1. Обчислення за трьома умовами

Напишіть програму для обчислення y залежно від x за своїм варіантом таблиці 1. Кожна з трьох формул застосовується лише для діапазону x, зазначеного в заголовку її стовпця.

1. Введіть невід'ємне ціле число x. За допомогою if/elif/else обчисліть одне значення y: у гілці if застосуйте першу формулу для $0 \le x < 2$, у гілці elif — другу для $2 \le x < 5$, у гілці else — третю для $x \ge 5$. Виведіть обчислене y з двома знаками після десяткової крапки.
2. Введіть невід'ємні цілі початок і кінець та додатний цілий крок, де початок ≤ кінець. За допомогою циклу for і range() повторіть обчислення з пункту 1 для кожного x заданого діапазону. Виведіть таблицю пар x, y з двома десятковими знаками для y. Кінець діапазону включайте лише тоді, коли він досягається заданим кроком.

Початкові дані коректні. Функції і обробка винятків не потрібні. У формулах x^2 означає квадрат; ділення — звичайне дійсне. Перевірте x = 0, 1, 2, 4, 5 та власний діапазон. Контрольна таблиця для початку 1, кінця 5 і кроку 1 має значення y при x = 1, 2, 3, 4, 5; вони наведені в останньому стовпці.

Таблиця 1 — Три гілки обчислення та контрольні значення

Варіант	$0 \le x < 2$	$2 \le x < 5$	$x \ge 5$	y при x = 1, 2, 3, 4, 5
1	x + 2	2x + 1	$x^2 - 5$	3.00; 5.00; 7.00; 9.00; 20.00.
2	2x	$x^2 + 1$	3x − 2	2.00; 5.00; 10.00; 17.00; 13.00.
3	5 − x	10 / (x + 1)	x + 5	4.00; 3.33; 2.50; 2.00; 10.00.
4	x^2	2x − 1	x / 2 + 4	1.00; 3.00; 5.00; 7.00; 6.50.

Варіант	$0 \le x < 2$	$2 \le x < 5$	$x \ge 5$	y при x = 1, 2, 3, 4, 5
5	$3x + 1$	$12 - 2x$	$x^2 / 5$	4.00; 8.00; 6.00; 4.00; 5.00.
6	4	$x(x + 1)$	$20 / x$	4.00; 6.00; 12.00; 20.00; 4.00.
7	$x / 2$	$(x - 1)^2$	$2x + 3$	0.50; 1.00; 4.00; 9.00; 13.00.
8	$6 - 2x$	$x + 3$	$(x - 2)^2$	4.00; 5.00; 6.00; 7.00; 9.00.
9	$2 + x^2$	$8 / x$	$2x - 4$	3.00; 4.00; 2.67; 2.00; 6.00.
10	$1 / (x + 1)$	$3x$	$x^2 - x$	0.50; 6.00; 9.00; 12.00; 20.00.
11	$2x + 5$	$15 - x^2$	$(x + 1) / 2$	7.00; 11.00; 6.00; -1.00; 3.00.
12	$x + 1$	$2(x - 2)^2$	$30 / (x + 1)$	2.00; 0.00; 2.00; 8.00; 5.00.
13	$3 - x^2$	$x(x - 1)$	$x + 10 / x$	2.00; 2.00; 6.00; 12.00; 7.00.
14	$2 / (x + 1)$	$x^2 / 2$	$4x - 7$	1.00; 2.00; 4.50; 8.00; 13.00.
15	$4x + 1$	$9 - x$	$(x - 1)^2 / 2$	5.00; 7.00; 6.00; 5.00; 8.00.

Завдання 2. Статистика введених чисел

Вводьте цілі числа окремими рядками до команди «стоп». Перед перевіркою команди застосуйте strip() і lower(); інші рядки містять коректні цілі числа.

1. Через while накопичуйте кількість і суму чисел, найменше і найбільше значення.
2. Після завершення виведіть ці показники і середнє з двома десятковими знаками. Якщо чисел немає, виведіть «Даних немає».

Не зберігайте всі числа у списку і не використовуйте sum(), min(), max(). Перше число врахуйте в усіх підсумках. Перевірте власні дані, один елемент і негайне завершення.

Таблиця 2 — Контрольна перевірка поточної статистики

Послідовність введення	Очікуваний результат	Що перевіряється
5, −2, 0, 7, стоп	Кількість 4, сума 10. Мінімум −2, максимум 7. Середнє 2.50.	Змішані числа
−5, −2, стоп	Кількість 2, сума −7. Мінімум −5, максимум −2. Середнє −3.50.	Усі числа від’ємні
4, стоп	Кількість 1, сума 4. Мінімум і максимум 4. Середнє 4.00.	Один елемент
« СТОП » першим рядком	«Даних немає».	Негайне завершення

Завдання 3. Аналіз слів та їхніх сусідів

Введіть речення без розділових знаків; поділіть його на слова через split(). Для порівнянь використовуйте нижній регістр, а у виведенні збережіть початкове написання.

1. За допомогою циклу for послідовно переберіть слова в порядку їх появи в реченні. Для кожного слова, що відповідає умові свого варіанта таблиці 3, виведіть його номер у введеному реченні та саме слово. Слова нумеруйте від 1.
2. Під час перебору підрахуйте кількість відібраних слів і суму їхніх довжин. Визначте найдовше серед них. Якщо кілька відібраних слів мають однакову найбільшу довжину, оберіть те, що трапилося першим у реченні.

Коли збігів немає, виведіть «Збігів немає», кількість і суму довжин 0. Найдовше слово в цьому випадку не виводьте. Окремого списку відібраних слів не створюйте. Слово без потрібного попереднього чи наступного сусіда не відбирайте. Перевірте порожній рядок, одне слово, повторення та власні дані. Контрольне речення — «код дані дані Python я код мова». У таблиці наведено приклад виведення відібраних слів із їхніми номерами та підсумки: кількість, суму довжин і найдовше слово.

Для варіанта 3 обидва правила застосуйте до того самого слова: (починається з «д» або закінчується на «а») і не коротше за попереднє. Для варіанта 15 контрольне речення вказано в його рядку.

Таблиця 3 — Правила аналізу слів і сусідніх елементів

Варіант	Умова відбору	Номер і слово	Підсумки
1	Довжина ≥ 4, слово містить «а» і довше за попереднє.	2: дані 7: мова	Кількість: 2. Сума довжин: 8. Найдовше слово: дані.
2	Довжина слова і його номер непарні. Слово коротше за наступне.	1: код 5: я	Кількість: 2. Сума довжин: 4. Найдовше слово: код.
3	(Починається з «д» або закінчується на «а») і не коротше за попереднє.	2: дані 3: дані 7: мова	Кількість: 3. Сума довжин: 12. Найдовше слово: дані.
4	Слово довше за попереднє.	2: дані 4: Python 6: код 7: мова	Кількість: 4. Сума довжин: 17. Найдовше слово: Python.
5	Слово коротше за наступне.	1: код 3: дані 5: я 6: код	Кількість: 4. Сума довжин: 11. Найдовше слово: дані.
6	Слово дорівнює попередньому.	3: дані	Кількість: 1. Сума довжин: 4. Найдовше слово: дані.
7	Слово дорівнює наступному.	2: дані	Кількість: 1. Сума довжин: 4. Найдовше слово: дані.
8	Перша літера дорівнює першій літері попереднього слова.	3: дані	Кількість: 1. Сума довжин: 4. Найдовше слово: дані.
9	Довжина дорівнює довжині наступного слова.	2: дані	Кількість: 1. Сума довжин: 4. Найдовше слово: дані.
10	Номер слова парний, довжина ≥ 3, слово відрізняється від попереднього.	2: дані 4: Python 6: код	Кількість: 3. Сума довжин: 13. Найдовше слово: Python.
11	Слово містить «о», не починається з «м» і не довше за наступне.	1: код 6: код	Кількість: 2. Сума довжин: 6. Найдовше слово: код.
12	Слово між двома сусідами і довше за обох.	4: Python	Кількість: 1. Сума довжин: 6. Найдовше слово: Python.
13	Остання літера дорівнює останній літері наступного слова.	2: дані	Кількість: 1. Сума довжин: 4. Найдовше слово: дані.

Варіант	Умова відбору	Номер і слово	Підсумки
14	Довжина парна, а попереднє слово має непарну довжину.	2: дані 7: мова	Кількість: 2. Сума довжин: 8. Найдовше слово: дані.
15	Попереднє і наступне слова мають однакову довжину.	Речення: «код дані кіт» 2: дані	Кількість: 1. Сума довжин: 4. Найдовше слово: дані.

Оформлення звіту

Звіт оформлюють за чинними вимогами курсу (документ «Вимоги до оформлення»).

Для кожного завдання у звіті подають формулювання, текст програми та скріншот результату в консолі.

Методичні рекомендації

Три гілки та граничні значення

Оператор if перевіряє першу умову, elif — наступну лише за хибності попередньої, else — решту випадків. Для меж 2 і 5 зручно перевірити x < 2, потім x < 5. Тоді x = 2 потрапляє у другу гілку, x = 5 — у третю. Декілька незалежних if можуть виконати більше ніж одну гілку.

Математичний запис 2x перекладають як 2 * x, x² — x ** 2, а |x| — abs(x). У Python ^ не є піднесенням до степеня. Дужки визначають порядок обчислень. Формат f"{value:.2f}" округлює тільки текст виведення; у змінній зберігайте початковий результат.

Порядок перевірок if/elif/else

Код Python

```
x = 4
if x < 3:
    result = x + 1
elif x < 7:
    result = 2 * x
else:
    result = x ** 2
print(result)
```

Результат виконання

```
8
```

Перша умова хибна, друга істинна, тому виконується лише друга гілка. Значення x = 3 теж належало б другій гілці, x = 7 — третій. Межі і формули у завданні інші.

Перебір числового діапазону

range(start, stop, step) повертає цілі значення, не включаючи stop. Щоб допустимий кінець включався, використайте range(start, finish + 1, step). range(1, 6, 2) дає 1, 3, 5; range(1, 5, 2) — 1, 3. Якщо крок не досягає кінця, додавати його окремо не потрібно.

Ті самі правила вибору гілки застосовують у разовому обчисленні та всередині циклу таблиці. Заголовок таблиці виводять перед циклом, пару x, y — на кожній ітерації.

Кінець range() не включається

Код Python

```
for value in range(1, 6, 2):
    print(value)
print("Коротший діапазон:")
for value in range(1, 5, 2):
    print(value)
```

Результат виконання

```
1
3
5
Коротший діапазон:
1
3
```

У першому range() верхня межа 6 дозволяє отримати 5; у другому stop = 5 виключає його. Крок 2 не означає, що довільний заданий кінець обов’язково з’явиться.

Поточні підсумки та завершення введення

while True разом із break дозволяє читати дані до команди. Спочатку перевіряйте команду, потім перетворюйте рядок через int(). Лічильник і суму почніть з 0. Мінімум і максимум установіть за першим фактичним числом, а не за 0: інакше набір лише додатних чисел дасть помилковий мінімум.

Наступні значення порівнюйте з поточними мінімумом і максимумом. Середнє обчислюйте після завершення як total / count, лише коли count > 0. Негайне введення команди не повинно спричиняти ділення на нуль.

Перша величина задає мінімум і максимум

Код Python

```
count = 0
for part in "4 -3 2".split():
    number = int(part)
    if count == 0:
        smallest = largest = number
    else:
        if number < smallest:
            smallest = number
        if number > largest:
            largest = number
    count += 1
print(count, smallest, largest)
```

Результат виконання

```
3 -3 4
```

У прикладі джерело — готовий рядок. Перше число враховується як обидві початкові межі, тому набір лише додатних або від’ємних чисел не потребує штучного нуля. У завданні додайте власні поточні підсумки до введення через while.

Команда перевіряється до перетворення числа

Код Python

```
while True:
    text = input("Число або кінець: ").strip()
    if text.lower() == "кінець":
        break
    print(int(text) * 2)
```

Результат виконання

```
Число або кінець: 3
6
Число або кінець: -1
-2
Число або кінець:  КІНЕЦЬ
```

break завершує цикл до int(), тому команда не перетворюється на число. Тут обчислюється подвійне значення; у завданні цикл накопичує статистику.

Позиції та сусідні слова

enumerate(words, start=1) дає пари позиція, слово. Якщо потрібні сусіди, перебирайте індекси через range(len(words)): попередній має індекс j − 1, наступний — j + 1. До читання сусіда перевірте j > 0 або j + 1 < len(words).

Логічний and перевіряє праву частину лише за істинності лівої. Тому j > 0 and len(words[j]) > len(words[j - 1]) безпечно порівнює довжини. На першому слові j − 1 без перевірки вкаже на останнє слово, а не на відсутнього сусіда.

Нормалізуйте слово через lower() для порівняння, проте друкуйте початковий words[j]. startswith() і endswith() перевіряють початок та кінець, in — наявність фрагмента. Парність позиції перевіряють через (j + 1) % 2. Для першого найдовшого оновлюйте результат лише за строго більшої довжини.

Індекс, позиція та безпечний сусід

Код Python

```
words = "я код дані".split()
for j in range(len(words)):
    if j > 0 and len(words[j]) > len(words[j - 1]):
        print(j + 1, words[j])
```

Результат виконання

```
2 код
3 дані
```

Перевірка j > 0 захищає доступ до попереднього слова. Індекс j починається з 0, початкова позиція j + 1 — з 1. Для наступного сусіда спочатку перевіряйте j + 1 < len(words).

Часті помилки

Не змішуйте індекс від 0 із позицією від 1. Не звертайтеся до сусідів перед перевіркою меж. Не рахуйте команду завершення як число. У статистиці і відборі явно обробляйте відсутність даних. Перевірки біля меж потрібні навіть тоді, коли інші приклади працюють.

Приклади виконання завдань

Приклад до завдання 1

Обчислимо навчальний показник: x + 1 для x < 3, 2x для 3 ≤ x < 6, x² для x ≥ 6. Разовий розрахунок і таблиця використовують однакові гілки.

Код Python

```python
x = int(input("Значення x: "))
if x < 3:
    result = x + 1
elif x < 6:
    result = 2 * x
else:
    result = x ** 2
print(f"Результат: {result:.2f}")
start = int(input("Початок: "))
finish = int(input("Кінець: "))
step = int(input("Крок: "))
print("x | результат")
for current in range(start, finish + 1, step):
    if current < 3:
        value = current + 1
    elif current < 6:
        value = 2 * current
    else:
        value = current ** 2
    print(f"{current} | {value:.2f}")
```

Приклад виконання в консолі

```
Значення x: 6
Результат: 36.00
Початок: 2
Кінець: 6
Крок: 2
x | результат
2 | 3.00
4 | 8.00
6 | 36.00
```

Пояснення до прикладу

input() зчитує x як текст, а int() перетворює його на ціле число. Розгалуження послідовно перевіряє умови й виконує лише одну формулу. Для x = 6 обидві умови хибні, тому else обчислює 6² = 36. Позначення :.2f у f-рядку задає виведення 36.00.

Після введення меж і кроку range(2, 7, 2) утворює значення 2, 4, 6. Верхня межа 7 не включається, проте дозволяє отримати заданий кінець 6. Для цих значень формули дають відповідно 3.00, 8.00 та 36.00.

У циклі формулу обирають за current, а не за початковим x. Значення 3 належить другій гілці, 6 — третій. Разове обчислення й таблиця використовують однакові умови та формули, змінюється лише поточне значення аргументу.

Приклад до завдання 2

До команди завершення порахуємо лише додатні числа. Цей приклад показує організацію введення; у завданні потрібно додатково накопичити статистику.

Код Python

```
positive = 0
while True:
    text = input("Число або стоп: ").strip()
    if text.lower() == "стоп":
        break
    value = int(text)
    if value > 0:
        positive += 1
print(f"Додатних чисел: {positive}")
```

Приклад виконання в консолі

```
Число або стоп: -4
Число або стоп: 7
Число або стоп: 0
Число або стоп: 2
Число або стоп:   СТОП
Додатних чисел: 2
```

Пояснення до прикладу

Змінна positive спочатку дорівнює 0 і зберігає кількість додатних чисел. Цикл while True повторює введення, доки не виконається break. strip() прибирає крайні пробіли, а lower() дозволяє розпізнати команду незалежно від регістру: « СТОП » теж завершує введення.

Команду перевіряють до int(), тому слово «стоп» не перетворюється на число. Інші рядки перетворюються на цілі значення. У показаному запуску лише 7 і 2 задовольняють умову value > 0, тому лічильник збільшується двічі. Числа −4 і 0 його не змінюють.

Підсумковий print() після циклу виконується один раз. Якщо відразу ввести команду завершення, результат буде 0. У завданні 2 таким самим перебором накопичуйте всі потрібні показники.

Приклад до завдання 3

Знайдемо слова, перша літера яких дорівнює останній літері попереднього. Це інша умова, ніж у варіантах, але вона показує перевірку сусіда і поточні підсумки.

Код Python

```
words = input("Речення: ").split()
count = 0
total_length = 0
longest = ""
for j in range(len(words)):
    if j > 0 and words[j][0].lower() == words[j - 1][-1].lower():
        print(f"{j + 1}: {words[j]}")
        count += 1
        total_length += len(words[j])
        if len(words[j]) > len(longest):
            longest = words[j]
print(f"Кількість: {count}; сума довжин: {total_length}")
if count:
    print(f"Найдовше: {longest}")
else:
    print("Збігів немає")
```

Приклад виконання в консолі

```
Речення: код дані ідея якість
2: дані
3: ідея
4: якість
Кількість: 3; сума довжин: 14
Найдовше: якість
```

Пояснення до прикладу

split() утворює список слів, а range(len(words)) перебирає їхні індекси від 0. Перевірка j > 0 пропускає перше слово, яке не має попереднього сусіда. Завдяки and порівняння літер виконується лише після цієї перевірки. lower() нормалізує літери для порівняння, проте виводиться початкове написання слова та його номер j + 1.

У реченні «код дані ідея якість» остання літера «код» збігається з першою літерою «дані». Так само утворюються пари «дані — ідея» та «ідея — якість». Тому відібрано слова з номерами 2, 3 і 4. Їхні довжини 4, 4 і 6 дають суму 14; найдовше слово — «якість».

Показники накопичуються під час відбору. Порівняння > зберігає перше найдовше слово за рівності довжин. Порожнє речення або відсутність збігів дають нульові підсумки й повідомлення «Збігів немає».

Контрольні питання

1. Чим if/elif/else відрізняється від трьох незалежних if?
2. Як перевірити значення точно на межі двох гілок?
3. Як записати квадрат і модуль у Python?
4. Чому stop у range() не включається?
5. Коли кінець діапазону не з’явиться в таблиці?
6. Чому команду завершення перевіряють до int()?
7. Як ініціалізувати мінімум і максимум?
8. Коли можна обчислювати середнє?
9. Як безпечно звертатися до сусідніх слів?
10. Як обрати перше найдовше слово за рівності довжин?