МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра "Захист інформації"

Написання програм з умовними виразами та циклами

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи № 4
з курсу «Програмування скриптовими мовами»
для студентів спеціальності
«Кібербезпека»

Затверджено
на засіданні кафедри
"Захист інформації"
протокол № 01 від 29.08.2024 р.

Львів – 2024

Написання програм з умовними виразами та циклами: Методичні вказівки до лабораторної роботи № 4 з курсу «Програмування скриптовими мовами» для студентів спеціальності «Кібербезпека» / Укл. *Я. Р. Совин* – Львів: Національний університет "Львівська політехніка", 2024. – 25 с.

Укладач: Я. Р. Совин, канд. техн. наук, доцент

Відповідальний за випуск: В. Б. Дудикевич, д.т.н., професор

Рецензенти: А. Я. Горпенюк, канд. техн. наук, доцент
Ю. Я. Наконечний, канд. техн. наук, доцент

Мета роботи – ознайомитись умовними операторами та циклічними конструкціями мови Python**.**

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Розглянемо оператори мови Python, що реалізують *умовні вирази* (conditional), де деякі інші оператори можуть бути виконані або невиконані в залежності від певних умов, і *цикли* (loop), де деякі інші оператори можуть бути виконані багатократно, знову ж таки в залежності від певних умов.

1.1. Умовний оператор if … else

Умовний оператор *if ... else* дозволяє в залежності від значення логічного виразу виконати окрему ділянку програми або, навпаки, не виконати її. Оператор має такий вигляд:

```
if <Логічний вираз 1>:
    <Блок, що виконується, якщо умова 1 істинна>
[elif <Логічний вираз 2>:
    <Блок, що виконується, якщо умова 2 істинна>
]
...
[else:
    <Блок, що виконується, якщо всі умови хибні>
]
```

Блоки всередині умовної інструкції виділяються однаковою кількістю пробілів (зазвичай чотирма). Кінцем блоку є інструкція, перед якою розташовано меншу кількість пробілів. У деяких мовах програмування логічний вираз заноситься в круглі дужки. У мові Python це робити необов'язково, але можна, оскільки будь-який вираз може бути розташований всередині круглих дужок. Тим не менше, круглі дужки слід використовувати тільки при необхідності розмістити умову на декількох рядках.

Для прикладу напишемо програму, яка перевіряє, чи є введене користувачем число парним чи ні. Після перевірки виводиться відповідне повідомлення.

```
x = int(input("Введіть число: "))

if x % 2 == 0:
    print (x, " - парне число")
else:
    print (x, " - непарне число")
```

Якщо блок складається з однієї інструкції, цю інструкцію можна розмістити в одному рядку з заголовком:

```
x = int(input("Введіть число: "))

if x % 2 == 0: print (x, " - парне число")
else:          print (x, " - непарне число")
```

У цьому випадку кінцем блоку є кінець рядка. Це означає, що можна розмістити відразу кілька інструкцій в одному рядку, розділяючи їх крапкою з комою:

```
x = int(input("Введіть число: "))

if x % 2 == 0: print(x, end = " "); print ("- парне число")
else:          print(x, end = " "); print ("- непарне число")
```

Так зробити можна, але використовувати подібну конструкцію не рекомендується, бо це порушує стрункість коду і погіршує його супровід надалі. Завжди розміщуйте інструкцію в окремому рядку, навіть якщо блок містить тільки одну інструкцію.

```
x = int(input("Введіть число: "))

if x % 2 == 0:
    print(x, end = " ");
    print ("- парне число")
else:
    print(x, end = " ");
    print ("- непарне число")
```

Часто в програмі потрібно перевірити більше двох можливих ситуацій: для таких випадків у Python передбачений синтаксис *if-elif-else*. Python виконує тільки один блок в ланцюжку *if-elif-else*. Всі умови перевіряються по порядку до тих пір, поки одна з них не дасть істинний результат. Далі виконується блок коду, який слідує за цією умовою, а всі решта перевірки Python пропускає. Розглянемо це на прикладі.

```
print("""Якою операційною системою ви користуєтеся?
1 - Windows 10
2 - Windows 8.1
3 - Windows 8
4 - Windows 7
5 - Windows XP
6 - Інша """)

os = input("Введіть відповідне число: ")

if os == "1":
    ("Ви вибрали: Windows 10")
elif os == "2":
    print( "Ви вибрали: Windows 8.1")
elif os == "3":
    print( "Ви вибрали: Windows 8")
elif os == "4":
    print( "Ви вибрали: Windows 7")
elif os == "5":
    print( "Ви вибрали: Windows XP")
elif os == "6":
    print( "Ви вибрали: інша")
elif not os:
    print( "Ви не ввели число")
else:
    print("Ми не змогли визначити вашу операційну систему")

input()
```

Один умовний оператор *if* можна вкласти в інший. В цьому випадку відступ вкладеної інструкції повинен бути в два рази більше:

```
print("""Якою операційною системою ви користуєтеся?
1 - Windows 10
2 - Windows 8.1
3 - Windows 8
4 - Windows 7
5 - Windows XP
6 - Інша """)

os = input("Введіть відповідне число: ")

if os != "":
    if os == "1":
        ("Ви вибрали: Windows 10")
    elif os == "2":
        print( "Ви вибрали: Windows 8.1")
    elif os == "3":
        print( "Ви вибрали: Windows 8")
    elif os == "4":
        print( "Ви вибрали: Windows 7")
    elif os == "5":
        print( "Ви вибрали: Windows XP")
    elif os == "6":
        print( "Ви вибрали: інша")
    else:
        print("Ми не змогли визначити вашу операційну систему")
else:
    print( "Ви не ввели число")

input()
```

Умовний оператор *if ... else* має ще один формат:

```
<Змінна> = <Якщо істина> if <Умова> else <Якщо не істина>
```

Приклад:

```
>>> print ("Yes" if 10 > 2 else "No")
Yes
>>> s = "Yes" if 10 > 2 else "No"
>>> s
'Yes'
>>> s = "Yes" if 10 > 20 else "No"
>>> s
'No'
```

1.2. Цикл for

Цикли дозволяють виконати одні й ті ж інструкції багаторазово. Цикл *for* застосовується для перебору елементів послідовності і має такий формат:

```
for <Поточний елемент> in <Послідовність>:
    <Інструкції всередині циклу>
[else:
    <Блок, що виконується, якщо не використовувався оператор break>
]
```

Тут присутні такі конструкції:

♦ <*Послідовність*> - об'єкт, що підтримує механізм ітерації: рядок, список,

кортеж, діапазон, словник та ін.;

♦ *<Поточний елемент>* - на кожній ітерації через цю змінну доступний черговий елемент послідовності або ключ словника;

♦ *<Інструкції всередині циклу>* - блок, який буде багаторазово виконуватися;

♦ якщо всередині циклу не використовувався оператор *break*, то після завершення виконання циклу буде виконано блок в інструкції *else*. Цей блок не є обов'язковим.

Приклад перебору букв в слові:

```
for s in "str":
    print(s, end = " ")
else:
    print("\nЦикл виконаний")
```

Результат виконання:

```
s t r
Цикл виконаний
```

Виведемо суму елементів 1 + 2 + … + *n:*

```
total = 0
n = 100

for i in range(1, n + 1):
    total += i
print("total =", total)
```

Результат:

```
total = 5050
```

Виведемо факторіал *n!* = 1 * 2 * … * *n*:

```
product = 1
n = 10

for i in range(1, n + 1):
    product *= i
print("product =", product)
```

Результат:

```
product = 3628800
```

Тепер виведемо кожен елемент списку і кортежу в окремому рядку.

```
for x in [1, 2, 3]:
    print(x)
for y in (1, 2, 3) :
    print(y)
```

Цикл *for* дозволяє також перебрати елементи словників, хоча словники і не є послідовностями. В якості прикладу виведемо елементи словника двома способами. Перший спосіб використовує метод *keys()*, який повертає об'єкт *dict_keys*, що містить всі ключі словника:

```
arr = {"x": 1, "y": 2, "z": 3}
print(arr.keys())

for key in arr.keys():
    print(key, arr[key])
```

Результат:

```
dict_keys(['x', 'y', 'z'])
```

```
x 1
y 2
z 3
```

У другому способі ми просто вказуємо словник як параметр - на кожній ітерації циклу буде повертатися ключ, за допомогою якого всередині циклу можна отримати значення, відповідне цьому ключу:

```
for key in arr:
    print(key, arr[key])
```

Результат:

```
x 1
y 2
z 3
```

Зверніть увагу на те, що елементи словника можуть виводитися в довільному порядку, а не в порядку, в якому вони були вказані при створенні об'єкту. Щоб вивести елементи в алфавітному порядку, слід впорядкувати ключі за допомогою функції *sorted():*

```
for key in sorted(arr):
    print(key, arr[key])
```

За допомогою циклу *for* можна перебирати складні структури даних. В якості прикладу виведемо елементи списку кортежів:

```
arr = [(1, 2), (3, 4)] # Список кортежів

for a, b in arr:
    print(a, b)
```

Результат:

```
1 2
3 4
```

Python дозволяє вкладати інструкції циклів один в одного – утворювати так звані *вкладені цикли* (nested loops):

```
outer = [1, 2]
inner = [3, 4]
for i in outer:      # Зовнішній цикл
    for j in inner: # Внутрішній (вкладений) цикл
        print("i =", i, "j =", j)
```

Результат:

```
i = 1 j = 3
i = 1 j = 4
i = 2 j = 3
i = 2 j = 4
```

У табл. 1 показано приклади вкладених циклів.

Таблиця 1

Вкладені цикли for з використанням range()

Вкладені цикли	Вивід

`for i in range(3):`
`  for j in range(4):`
`  print("*", end = "")`
`  print()`
	****
	****
	****

`for i in range(4):`
`  for j in range(3):`
`  print("*", end = "")`
	***
	***
	***

`    print()`	***

`for i in range(4):`
`    for j in range(i + 1):`
`        print("*", end = "")`
`    print()`
	*
	**
	***
	****

`for i in range(3):`
`    for j in range(5):`
`        if i % 2 == j % 2:`
`            print("*", end = "")`
`        else:`
`            print(" ", end = "")`
`    print()`
	* * *
	 * *
	* * *

1.3. Функції range() та enumerate()

Досить часто на практиці, особливо при роботі з циклами необхідно отримати *діапазон* цілих чисел (рис. 1), наприклад, щоб згенерувати індекси елементів послідовності.

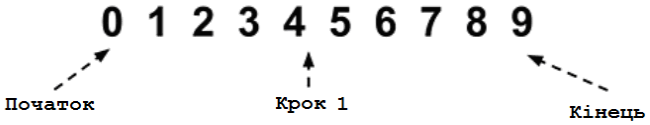

Рис. 1. Діапазон цілих чисел

Для вирішення цієї задачі в Python передбачена функція *range()*. Функція *range()* має наступний формат:

```
range([<Початок>,]<Кінець>[,<Крок>])
```

Перший параметр задає початкове значення. Якщо параметр <*Початок*> не вказано, то використовується 0. У другому параметрі вказується кінцеве значення. Слід зауважити, що це значення не входить в значення, що повертаються. Якщо параметр <*Крок*> не вказано, то використовується значення 1:

```
>>> range(0, 10, 1)
range(0, 10)
>>> idx = range(0, 10, 1)
>>> idx
range(0, 10)
>>> for i in range(0, 10, 1):
     print(i, end = ' ')
0 1 2 3 4 5 6 7 8 9
>>> for i in range(10):
     print(i, end = ' ')
0 1 2 3 4 5 6 7 8 9
>>> for i in range(2, 20, 2):
     print(i, end = ' ')
2 4 6 8 10 12 14 16 18
```

Функція повертає діапазон - особливий об'єкт, що підтримує ітераційний протокол. За допомогою діапазону всередині циклу *for* можна отримати значення поточного елемента. Як приклад помножимо кожен елемент списку на 2:

```
arr = [1, 2, 3]

for i in range(len(arr)):
```

```
    arr[i] *= 2
print(arr)
```

Результат:

```
[2, 4, 6]
```

У цьому прикладі ми отримуємо кількість елементів списку за допомогою функції *len()* і передаємо результат в функцію *range()*. У підсумку остання поверне діапазон значень від 0 до *len(arr) - 1*. На кожній ітерації циклу через змінну *i* доступний поточний елемент з діапазону індексів. Щоб отримати доступ до елемента списку, вказуємо індекс всередині квадратних дужок. Множимо кожен елемент списку на 2, а потім виводимо результат за допомогою функції *print()*.

Функція *range()* для економії місця зберігає в пам'яті тільки початок, кінець і крок діапазону. Це корисно при використанні циклів для перебору дуже великих послідовностей. Наприклад, замість того щоб будувати список з 10 мільйонів елементів, що потребує серйозних витрат пам'яті, можна скористатися викликом *range(10000000)*, який займе лише малу частку пам'яті.

Розглянемо кілька прикладів використання функції *range()*:

♦ Виведемо числа від 1 до 100:

```
for i in range(1, 101): print(i)
```

♦ Можна не тільки збільшувати значення, але і зменшувати його. Виведемо всі числа від 100 до 1:

```
for i in range(100, 0, -1): print(i)
```

♦ Можна також змінювати значення не тільки на одиницю. Виведемо всі парні числа від 1 до 100:

```
for i in range(2, 101, 2): print(i)
```

Щоб перетворити повернутий функцією *range()* діапазон в список чисел, слід передати цей діапазон в функцію *list()*:

```
>>> obj = range(len([1, 2, 3]))
>>> obj
range(0, 3)
>>> obj[0], obj[1], obj[2]
(0, 1, 2)
>>> obj[0:2] # Отримання зрізу
range(0, 2)
>>> i = iter(obj)
>>> next(i), next(i), next(i) # Доступ з допомогою ітераторів
(0, 1, 2)
>>> list(obj) # Перетворення діапазону в список
[0, 1, 2]
>>> 1 in obj, 7 in obj # Перевірка входження значення
(True, False)
```

Діапазон підтримує два корисних методи:

♦ *index(<Значення>)* - повертає індекс елемента, що має вказане значення. Якщо значення не входить в діапазон, генерується виключення *valueError*:

```
>>> obj = range(1, 5)
>>> obj.index(1), obj.index(4)
(0, 3)
>>> obj.index (5)
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    obj.index (5)
```

```
ValueError: 5 is not in range
```

♦ *count(<Значення>)* - повертає кількість елементів із зазначеним значенням. Якщо елемент не входить в діапазон, повертається значення 0:

```
>>> obj = range(1, 5)
>>> obj.count(1), obj.count(10)
(1, 0)
```

Функція *enumerate(<Об'єкт>[, start=0])* на кожній ітерації циклу *for* повертає кортеж з індексу і значення поточного елементу. З допомогою необов'язкового параметру *start* можна задати початкове значення індексу. Як приклад помножимо на 2 кожен елемент списку, який містить парне число:

```
arr = [1, 2, 3, 4, 5, 6]

for i, elem in enumerate(arr):
    if elem % 2 == 0:
        arr[i] *= 2
print(arr)
```

Результат:

```
[1, 4, 3, 8, 5, 12]
```

Функція *enumerate()* не створює список, а повертає ітератор. За допомогою функції *next()* можна обійти всю послідовність. Коли перебір буде закінчений, генерується виняток *StopIteration*:

```
>>> arr = [1, 2]
>>> obj = enumerate(arr, start = 2)
>>> next(obj)
(2, 1)
>>> next(obj)
(3, 2)
>>> next(obj)
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    next(obj)
StopIteration
```

Таким чином у табл. 2 зібрано типові приклади спільного застосування циклу *for* та діапазонів, що генеруються функцією *range()*.

Таблиця 2

Цикл for з використанням range()

Цикл	Значення i
for i in range(6):	0, 1, 2, 3, 4, 5
for i in range(10, 16):	10, 11, 12, 13, 14, 15
for i in range(0, 9, 2):	0, 2, 4, 6, 8
for i in range(5, 0, -1):	5, 4, 3, 2, 1

1.4. Цикл while

Інструкція *for* використовується, якщо наперед відомо, скільки ітерацій необхідно виконати. Якщо наперед кількість ітерацій невідома, то застосовується інструкція *while*.

Виконання інструкцій в циклі *while* триває до тих пір, поки логічний вираз є істинний. Цикл *while* має такий вигляд:

```
<Задання початкового значення для змінної-лічильника>
while <Умова>:
     <Інструкції>
     <Приріст значення в змінній-лічильнику>
[else:
     <Блок, що виконується, якщо не використовувався оператор break>
]
```

Послідовність роботи циклу *while*:

1. Змінній-лічильнику присвоюється початкове значення.

2. Перевіряється умова, і якщо вона істинна, то виконуються інструкції всередині циклу, інакше виконання циклу завершується.

3. Змінна-лічильник змінюється на величину, зазначену в параметрі *<Приріст>*.

4. Перехід до пункту 2.

5. Якщо всередині циклу не використовувався оператор *break*, то після завершення виконання циклу буде виконано блок в інструкції *else*. Цей блок не є обов'язковим.

Виведемо всі числа від 1 до 100, використовуючи цикл *while*.

```
i = 1           # <Початкове значення>
while i < 101: # <Умова>
    print(i)    # <Інструкції>
    i += 1      # <Приріст>
```

Увага! Якщо *<Приріст>* не вказано, цикл буде виконуватися нескінченно. Щоб перервати нескінченний цикл, слід натиснути комбінацію клавіш *<Ctrl>+<C>*. В результаті генерується виняток *KeyboardInterrupt*, і виконання програми зупиняється. Потрібно враховувати, що перервати таким чином можна тільки цикл, який виводить дані

Виведемо всі числа від 100 до 1:

```
i = 100
while i:
    print(i)
    i -= 1
```

Зверніть увагу на умову - воно не містить операторів порівняння. На кожній ітерації циклу ми віднімаємо одиницю зі значення змінної-лічильника. Як тільки значення дорівнюватиме 0, цикл зупиниться, оскільки число 0 в логічному контексті еквівалентно значенню *False*.

За допомогою циклу *while* можна перебирати і елементи різних структур. Але в цьому випадку слід пам'ятати, що цикл *while* працює повільніше циклу *for*. В якості прикладу помножимо кожен елемент списку на 2:

```
arr = [1, 2, 3]
i, count = 0, len(arr)

while i < count:
    arr[i] *= 2
    i += 1
print(arr)
```

Результат:

```
[2, 4, 6]
```

1.5. Оператори continue та break

Оператор *continue* дозволяє перейти до наступної ітерації циклу до завершення виконання всіх інструкцій всередині циклу. Як приклад виведемо всі числа від 1 до 100, крім чисел від 5 до 10 включно:

```
for i in range(1, 101):
    if 4 < i < 11:
        continue # Переходимо на наступну ітерацію циклу
    print(i)
```

Оператор *break* дозволяє перервати виконання циклу достроково. Для прикладу виведемо всі числа від 1 до 100 ще одним способом:

```
i = 1
while True:
    if i > 100: break # Перериваємо цикл
    print(i)
    i += 1
```

Тут ми в умові вказали значення *True*. У цьому разі вирази всередині циклу стануть виконуватися нескінченно. Однак використання оператора *break* перериває виконання циклу, як тільки він буде виконаний 100 разів.

Цикл *while* спільно з оператором *break* зручно використовувати для отримання не визначеної заздалегідь кількості даних від користувача. Як приклад підсумуємо довільну кількість чисел:

```
print("Введіть слово 'stop' для отримання результату")
summa = 0

while True:
    x = input("Введіть число: ")
    if x == "stop":
        break                       # Вихід з циклу
    x = int(x)                      # Перетворюємо рядок в число
    summa += x

print("Сума чисел рівна: ", summa)
input()
```

Процес введення трьох чисел і отримання суми виглядає так (значення, введені користувачем, тут виділені напівжирним шрифтом):

```
Введіть число: 1
Введіть число: 4
Введіть число: 9
Введіть число: stop
Сума чисел рівна:  14
```

2. ЗАВДАННЯ

2.1. Домашня підготовка до роботи

1. Вивчити теоретичний матеріал.

2.2. Виконати в лабораторії

1. Написати програму табулювання функції (див. табл. 3), що вибирається залежно від значення аргументу, на проміжку **[*a, b*]** з кроком табуляції ***h***. При табулюванні має виводитися аргумент ***x***, значення функції ***y*** з точністю 4 знаки після коми. Ширина полів аргументу і значення функції має бути фіксована і вирівняна.
2. Написати програму табулювання функції, представленої рядом (див. табл. 4), на інтервалі **[*a, b*] з** кроком табуляції ***h*** та абсолютною похибкою ***d***. Оцінку похибки здійснювати за значенням модуля чергового члена ряду. При табулюванні має виводитися аргумент ***x***, значення функції ***y*** та абсолютна похибка ***d*** з точністю 5 знаків після коми. Ширина полів аргументу, значення функції і похибки має бути фіксована і вирівняна.
3. Написати програму валідації введеного паролю. Програма не повинна використовувати регулярні вирази, списки, множини, словники, функції, класи чи сторонні бібліотеки окрім *colorama*. Користувач повинен ввести пароль, програма має перевірити наявність у ньому лише заданих типів символів у вказаних пропорціях і з дотримання додаткових правил згідно варіанту у табл. 5 і вивести інформацію про результати перевірки у форматі як показано на рис. 1.

```
Введіть пароль довжиною не менше 12 символів.
Вимоги до паролю:
1.  Маленькі латинські літери
2.  Великі латинські літери
3.  Цифри
4.  Спеціальні символи !@#$_-%^&*
5.  Не менше 3 і не більше 5 маленьких латинських літер
6.  Не менше 3 і не більше 5 великих латинських літер
7.  Не менше 2 і не більше 4 цифр
8.  Не менше 2 і не більше 4 спеціальних символів
9.  Не більше 3 однакових спеціальних символів
10. Не більше 3 однакових маленьких латинських літер підряд
> 67n!!!!aVFk
Довжина не менше 12 символів - FAIL!
Пароль містить лише допустимі символи - FAIL!
Маленькі латинські літери - FAIL!
Великі латинські літери - FAIL!
Цифри - OK!
Спеціальні символи - OK!
Не більше 3 однакових спеціальних символів - FAIL!
Не більше 3 однакових маленьких латинських літер підряд - OK!

Пароль не валідний!

Введіть пароль довжиною не менше 12 символів.
Вимоги до паролю:
1.  Маленькі латинські літери
2.  Великі латинські літери
3.  Цифри
4.  Спеціальні символи !@#$_-%^&*
5.  Не менше 3 і не більше 5 маленьких латинських літер
6.  Не менше 3 і не більше 5 великих латинських літер
7.  Не менше 2 і не більше 4 цифр
8.  Не менше 2 і не більше 4 спеціальних символів
9.  Не більше 3 однакових спеціальних символів
10. Не більше 3 однакових маленьких латинських літер підряд
> *6sss4SD*5ZZ*
Довжина не менше 12 символів - OK!
Пароль містить лише допустимі символи - OK!
Маленькі латинські літери - OK!
Великі латинські літери - OK!
Цифри - OK!
Спеціальні символи - OK!
Не більше 3 однакових спеціальних символів - OK!
Не більше 3 однакових маленьких латинських літер підряд - OK!

Пароль валідний!
```

Номер варіанту відповідає номеру в списку групи.

3. ЗМІСТ ЗВІТУ

1. Мета роботи.
2. Повний текст завдання згідно варіанту.
3. Лістинг програми.
4. Результати роботи програм (у текстовій формі та скріншот).
5. Висновок.

Табл. 3

№ п/п	Функції	Умови для вибору функцій	Крок, h	Інтервал, [a,b]
1	$\cos(\sqrt{x})$ $\mathrm{ctg}(x^2)$ $\mathrm{arctg}(x^3)$	$x < 0.6$ $0.6 \le x < 0.7$ $x \ge 0.7$	0.02	[0.5, 0.9]
2	$\ln(x^3)$ $\frac{1}{\lvert\sin(x)\rvert}$ $\sec(1/x)$	$x \le 3$ $3 < x < 3.5$ $x \le 4$	0.2	[2, 4]
3	$\mathrm{cosec}(x^2)$ $x + \ln(\sqrt{x^7})$ $\lg(e^x + 4)$	$x < 4.5$ $4.5 \le x < 5$ $x \ge 5$	0.2	[4, 6]
4	$\cos(\ln(x^2))$ $\sec(x^4)$ $\mathrm{tg}(\sin(x))$	$x < 2.5$ $2.5 \le x \le 3.5$ $x > 3.5$	0.2	[2, 4]
5	$\log_5(3x+1)$ $x^{\cos(x)}$ $\mathrm{cosec}(\ln(x))$	$x < 0.2$ $0.2 \le x < 0.4$ $x \ge 0.4$	0.05	[0.1, 0.7]
6	$e^{(x-\sin(x))}$ $\mathrm{tg}(\lvert\ln(x)\rvert)$ $\mathrm{arctg}(x^7)$	$x < 0.6$ $0.6 \le x < 07$ $x \ge 0.7$	0.02	[0.5, 0.8]
7	$\ln(x) - \cos(x)$ $\mathrm{ctg}(e^x + 3)$ $(5x)^x$	$x < 4$ $4 \le x < 5$ $x \ge 5$	0.2	[3, 6]

8	$\operatorname{cosec}(e^x)$ $\sec(\ln(x))$ $\sin(\ln(x))$	$x < 2$ $2 \le x < 3$ $x \ge 3$	0.2	[1.5, 3.5]
9	$\sec(x \cdot \cos(x))$ $x^3 + 4$ $\lg(e^x)$	$x < 4.5$ $4.5 \le x < 6$ $x \ge 6$	0.2	[4, 7]
10	$\cos(x) + \operatorname{tg}(x)$ $\operatorname{ctg}(x) + \sin(x)$ $(x \cdot \ln(x))^3$	$x < 2.3$ $2.3 \le x < 2.7$ $x \ge 2.7$	0.1	[2, 3]
11	$\log_3(x + \sin(x))$ $\lg(e^x + 4)$ $\ln(\lg(x))$	$x < 4$ $4 \le x < 5$ $x \ge 5$	0.2	[3, 6]
12	$\cos(\sqrt{x^3})$ $\operatorname{arctg}(e^x)$ $\sin^5(\ln(x))$	$x < 1$ $1 \le x < 1.5$ $x \ge 1.5$	0.2	[0.5, 2]
13	$-\ln(x^2) + e^x$ $\operatorname{ctg}(x^2 + 4)$ $\operatorname{tg}(x^2 + 1)$	$x < 0.3$ $0.3 \le x < 0.4$ $x \ge 0.4$	0.02	[0.2, 0.5]
14	$\lvert \ln(x) \rvert^7$ $\operatorname{ctg}(x + x^3)$ $\log_5(\sin(x))$	$x < 0.5$ $0.5 \le x < 0.7$ $x \ge 0.7$	0.02	[0.4, 0.8]
15	$(x^2 - 1)^{(x-7)}$ $\frac{1}{\sin(x) + \lvert \cos(x) \rvert}$ $\ln(e^x + 4)$	$x \le 8$ $8 < x \le 9$ $x > 9$	0.2	[7.5, 10]
16	$\cos(x^{0.3})$ $\sqrt{x^3 + \lg(x)}$ $\operatorname{ctg}(x^2)$	$x < 3$ $3 \le x < 4$ $x \ge 4$	0.2	[2, 5]
17	$\operatorname{cosec}(\cos(x^2))$ $\cos(\sin(x))$ $\sin(\sec(x))$	$x < 0.5$ $0.5 \le x < 0.7$ $x \ge 0.7$	0.05	[0.3, 0.9]
18	$\operatorname{arctg}(x^3)$ $tg(x + \ln(\lvert x \rvert))$ $ctg(x^2)$	$x \le -0.7$ $-0.7 < x \le -0.6$ $x \ge -0.6$	0.05	[-0.9, -0.4]
19	$\ln(\lvert \lg(x) + \log_3(x) \rvert)$ $\cos(\sin(x^2))$ $\sqrt[7]{x^3 + 0.5}$	$x \le 0.4$ $0.4 < x < 0.6$ $x \ge 0.6$	0.05	[0.3, 0.9]

20	$arctg(1/x)$ $tg(x+\log_4(x))$ $\frac{1}{1+\ln(x)}$	$x<1$ $1\le x<3$ $x\ge 3$	0.3	[0.3, 3.5]
21	$\frac{1}{\sin(1/x)+4}$ $\frac{1}{x^2+\ln(x)}$ $tg((x-3)^3)$	$x<4$ $4\le x<5$ $x\ge 5$	0.3	[3, 6]
22	$ctg(x+\cos ec(x^{-2}))$ $\lg(\ln(x)+\log_3(x))$ $\cos(5x^2)$	$x<3$ $3\le x<4$ $x\ge 4$	0.2	[2, 5]
23	$\log_5(5+\lvert\log_4(\lvert\log_3(x)\rvert)\rvert)$ $\frac{1}{x^2+16}$ $\ln(x)+\cos(x)$	$x<5$ $5\le x<7$ $x\ge 7$	0.5	[3, 8]
24	$\lg(\lvert\ln(x)+\sec(x)\rvert)$ $ctg(x+\ln(x))$ $\frac{1}{16-x^2}$	$x<3$ $3\le x\le 4$ $x>4$	0.2	[2, 5]
25	$\lg(x\cdot\ln(x)+\sin(x))$ $\log_3(\sin(x)+4)$ $\frac{1}{16+\sec(x)}$	$x<7$ $7\le x<8$ $x\ge 8$	0.2	[6, 9]

26	$\sin(x^2)+\cos(x^2)$ $\log_7(x+\mathrm{tg}(x))$ $\frac{1}{160+x^2}$	$x < 0.8$ $0.8 \le x < 1$ $x \ge 1$	0.5	[0,6, 1,2]
27	$\frac{1+\cos(x)}{1-\sin(x^2)}$ $\mathrm{arctg}(\ln(x+\sin(x)))$ $\frac{1}{x^2}+\frac{1}{x^3}+\frac{1}{x^4}$	$x < 4$ $4 \le x < 5$ $x \ge 5$	0.2	[3, 6]
28	$\sqrt{\lvert\mathrm{tg}(x+\sin(x))\rvert}$ $\frac{1}{x^3+\sin(x^2)}$ $e^{\mathrm{tg}(x+2)}$	$x < 2$ $2 \le x < 2.5$ $x \ge 2.5$	0.1	[1.5, 3]
29	$\frac{1}{\sqrt{\lvert\mathrm{tg}(x^2+5)\rvert}}$ $\cos(\ln(x+\sqrt{x}))$ $\log_5(\cos^2(x+7))$	$x < 2$ $2 \le x < 3$ $x \ge 3$	0.5	[0, 5]

30	$-\operatorname{cosec}\left(\sqrt{x^2+e^x}\right)$ $\cos\left(x^{\ln(x)}\right)$ $\operatorname{ctg}\left(x^3+11.5\right)$	$x<5$ $7\le x<5$ $x\ge 7$	0.3	[3, 9]
31	$\log_5(x)-\cos(x-7)$ $\frac{\operatorname{arctag}(x)}{\sin(\ln(x))}$ $\frac{\sin(x^2)}{x^3}+\operatorname{tg}(2+x)$	$x<2.8$ $2.8\le x<3.4$ $x\ge 3.4$	0.1	[2, 4]
32	$\lg\left(\left\|\cos(x)+e^{x-2}\right\|\right)$ $\sin\left(x+2x^3\right)-\cos\left(x-2x^2\right)$ $\frac{\arctan(x)}{\sqrt{x^2-\sin(x)}}$	$x<2$ $2\le x<3$ $x\ge 3$	0.2	[1, 4]
33	$\sqrt[7]{\sin(x)\ln(x)+5}$ $\frac{\sin\left(x+\sqrt{x}\right)}{\ln\left(x^3+4\right)}$ $(x+1)^{\sin(x+1)}$	$x<0.5$ $0.5\le x<0.6$ $x\ge 0.6$	0.05	[0.2, 0.9]
34	$\frac{x+1}{\operatorname{tg}\left(\ln\left(x^2+3\right)\right)}$ $\sqrt[3]{\frac{\sin(x+5)}{2x-x^3}}$ $-\log_4\left(\left\|\frac{\sin(x)}{x-4}\right\|\right)$	$x<5$ $5\le x<6$ $x\ge 6$	0.2	[4, 7]
35	$\sin\left(x^3\right)+\frac{e^{2-x}}{102.1}$ $tg\left(\frac{x^3}{2.7}\right)+\sin\left(\sqrt{x}\right)$ $\frac{1}{x^2}-\frac{1}{x^3}-\frac{1}{x^4}$	$x<5$ $5\le x<7$ $x\ge 7$	0.3	[3, 9]

Табл. 4

№ п/п	**Функція**	**Інтервал, [a, b]**	**Крок, h**	**Похибка, d**
1	$\sum_{k=1}^{\infty}\frac{x+2}{k(k+2)}$	[0.5, 0.7]	0.02	0.001
2	$\sum_{k=1}^{\infty}kx^k$	[0.1, 0.6]	0.05	0.001
3	$\sum_{k=1}^{\infty}\frac{1}{k}\operatorname{tg}\left(\frac{x}{2^k}\right)$	[3, 4]	0.1	0.001

4	$\sum_{k=0}^{\infty} \frac{1}{4k+3} x^{(4k+3)}$	[0.2, 0.3]	0.01	10^{-6}
5	$\sum_{k=1}^{\infty} \frac{1}{2^k} \sin\left(\frac{x}{2^k}\right)$	[1.1, 2]	0.1	0.001
6	$\sum_{k=1}^{\infty} \frac{(-1)^k x}{k(k+1)} \sin(2k+1)$	[-1, 1]	0.5	0.001
7	$\sum_{k=2}^{\infty} \frac{(-1)^k k}{k^2-1} \sin(kx)$	[-1, 1]	0.2	0.001
8	$\sum_{k=2}^{\infty} \frac{(-1)^k \cos(kx)}{k^2-1}$	[-1, -0.9]	0.01	0.001
9	$\sum_{k=0}^{\infty} \frac{x}{(2k+1)^3} \sin(2k+1)$	[-1, 1]	0.1	0.001
10	$\sum_{k=0}^{\infty} \frac{x}{(2k-1)(2k+3)} \cos(2k+1)$	[0.1, 1]	0.1	0.001
11	$\sum_{k=0}^{\infty} \frac{(-1)^k x^{(2k+3)}}{(2k+1)(2k+3)}$	[0, 1]	0.1	0.0001

12	$\sum_{k=1}^{\infty}\frac{(-1)^k\cos^4(2^k x)}{2^{2k}}$	[1, 2]	0.1	0.001
13	$\sum_{k=1}^{\infty}\frac{(-1)^k\sin(kx)}{k}$	[3, 4]	0.1	0.001
14	$1+\sum_{n=1}^{\infty}\frac{(m-n+1)}{n!}x^n$, m=20	[0.1, 0.5]	0.05	0.001
15	$1+\sum_{n=1}^{\infty}\frac{(m+n-1)}{n!}x^n$, m=3	[0.1, 0.5]	0.05	0.001
16	$2\sum_{n=1}^{\infty}\frac{(x-1)^{(2n+1)}}{(2n+1)(x+1)^{(2n+1)}}$	[1, 1.2]	0.02	10^{-6}
17	$\sum_{n=1}^{\infty}(-1)^{(n+1)}\frac{(x-1)^n}{n}$	[1, 1.5]	0.05	10^{-5}
18	$\sum_{n=1}^{\infty}\frac{(x-1)^n}{nx^n}$	[1, 1.5]	0.05	10^{-5}
19	$\sum_{n=1}^{\infty}(-1)^{(n+1)}\frac{x^n}{n}$	[0, 0.5]	0.05	0.001
20	$-\sum_{n=1}^{\infty}\frac{x^n}{n}$	[-0.5, 0]	0.05	0.001

21	$1+\sum_{n=1}^{\infty}\frac{(2n-1)}{(2n)(2n+1)}x^{n}$	[0.5, 0.9]	0.05	0.001
22	$\frac{\pi}{2}-x-\sum_{n=1}^{\infty}\frac{(2n-1)}{(2n)(2n+1)}x^{(2n+1)}$	[0.5, 0.9]	0.05	0.001
23	$\sum_{n=1}^{\infty}\frac{x^{2n}-1}{(2n-1)!}$	[0.1, 0.2]	0.01	0.001
24	$1+\sum_{n=1}^{\infty}\frac{(-1)^{n}(m-n+1)}{n!}x^{n}$, m=20	[0, 0.5]	0.05	0.001
25	$1+\sum_{n=1}^{\infty}\frac{(-1)^{n}(m+n-1)}{n!}x^{n}$, m=9	[0, 0.5]	0.05	0.001
26	$\sum_{n=1}^{\infty}\frac{\cos(xn)}{(n+2)(n+3)}$	[1, 5]	0.5	0.001
27	$1+\sum_{n=1}^{\infty}\frac{x}{(3n-1)^{3}}\sin(n+2)$	[0.1, 0.6]	0.05	0.001
28	$\sum_{n=1}^{\infty}\frac{(-1)^{n-1}(n+1)}{n!}\sin(x)$	[-1, 1]	0.2	0.001
29	$\sum_{n=1}^{\infty}\frac{\cos^{3}(x)}{n!}$	[0.1, 0.5]	0.05	0.0001
30	$\sum_{k=1}^{\infty}\frac{x}{(2k+1)(2k+2)}\sin(k+1)$	[0.5, 1]	0.05	0.00001
31	$\sum_{k=1}^{\infty}\frac{\cos^{2}(x)}{(k+2)^{2}}$	[1, 2]	0.1	0.0001
32	$\sum_{k=1}^{\infty}\frac{(-1)^{k}\sin(kx)}{k}$	[3, 4]	0.1	0.001
33	$1+\sum_{n=1}^{\infty}\frac{(2n-1)}{(2n)(2n+1)}x^{n}$	[0.5, 0.9]	0.05	0.001
34	$\sum_{n=1}^{\infty}(-1)^{(n+1)}\frac{(x-1)^{n}}{n}$	[1, 1.5]	0.05	10^{-5}

У якості наборів символів можуть виступати:

upp_char = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
low_char = "abcdefghijklmnopqrstuvwxyz"
num_char = "0123456789"
spc_char = "!@#$%^&*_-"

Додаткові правила для формування паролів:

1. Не менше 2 маленьких латинських літер
2. Не менше 3 маленьких латинських літер
3. Не менше 4 маленьких латинських літер
4. Не менше 5 маленьких латинських літер
5. Не менше 2 великих латинських літер
6. Не менше 3 великих латинських літер
7. Не менше 4 великих латинських літер
8. Не менше 5 великих латинських літер
9. Не менше 2 цифр
10. Не менше 3 цифр
11. Не менше 2 спеціальних символів
12. Не менше 3 спеціальних символів
13. Не більше 4 маленьких латинських літер
14. Не більше 5 маленьких латинських літер
15. Не більше 6 маленьких латинських літер
16. Не більше 7 маленьких латинських літер
17. Не більше 4 великих латинських літер
18. Не більше 5 великих латинських літер
19. Не більше 6 великих латинських літер
20. Не більше 4 цифр
21. Не більше 5 цифр
22. Не більше 3 спеціальних символів
23. Не більше 4 спеціальних символів
24. Не більше 2 однакових маленьких латинських літер
25. Не більше 3 однакових маленьких латинських літер
26. Не більше 2 однакових великих латинських літер
27. Не більше 3 однакових великих латинських літер
28. Не більше 2 однакових цифр
29. Не більше 2 однакових спеціальних символів
30. Не більше 3 однакових спеціальних символів
31. Не більше 3 однакових маленьких латинських літер підряд
32. Не більше 3 однакових великих латинських літер підряд
33. Не більше 3 однакових цифр підряд
34. Не більше 3 однакових спеціальних символів підряд

Табл. 5

Варіанти завдань

Варіант	Довжина, символів	Набір символів	Додаткові правила
1.	12	low_char + spc_char + upp_char	3, 15, 11, 23, 7, 18, 25, 32
2.	15	low_char + num_char + upp_char	4, 16, 9, 21, 8, 19, 27, 33
3.	10	low_char + num_char + spc_char + upp_char	1, 14, 10, 20, 11, 22, 5, 17, 28, 34
4.	9	spc_char + upp_char	12, 23, 6, 17, 26, 32
5.	12	low_char + num_char + spc_char + upp_char	2, 14, 9, 20, 11, 23, 6, 18, 30, 31
6.	10	low_char + num_char + spc_char	1, 13, 10, 20, 12, 22, 29, 33
7.	14	low_char + num_char + spc_char + upp_char	3, 15, 10, 21, 11, 22, 7, 18, 29, 31
8.	9	low_char + upp_char	3, 14, 7, 18, 24, 32
9.	8	num_char + spc_char	9, 21, 11, 23, 30, 33
10.	9	low_char + spc_char	2, 15, 12, 23, 25, 34
11.	8	num_char + spc_char + upp_char	9, 20, 11, 22, 5, 17, 26, 33
12.	11	low_char + spc_char + upp_char	2, 13, 12, 23, 7, 18, 25, 32
13.	14	low_char + num_char + spc_char + upp_char	3, 16, 10, 21, 11, 23, 7, 19, 27, 34
14.	10	low_char + spc_char + upp_char	3, 15, 11, 22, 6, 18, 30, 31
15.	11	low_char + num_char + spc_char + upp_char	1, 13, 9, 20, 11, 22, 5, 17, 28, 32
16.	13	low_char + num_char + spc_char + upp_char	4, 15, 9, 21, 12, 23, 5, 18, 24, 34
17.	13	low_char + num_char + spc_char + upp_char	3, 16, 10, 20, 11, 22, 6, 19, 29, 33
18.	11	low_char + upp_char	2, 14, 6, 18, 25, 32
19.	12	low_char + num_char + spc_char + upp_char	2, 13, 9, 20, 11, 22, 5, 17, 27, 31
20.	11	low_char + spc_char + upp_char	2, 13, 12, 23, 6, 18, 26, 34
21.	6	num_char + spc_char	9, 20, 11, 23, 28, 34
22.	14	low_char + num_char + spc_char + upp_char	3, 15, 10, 21, 12, 23, 6, 18, 24, 32
23.	9	low_char + num_char + spc_char + upp_char	1, 14, 9, 21, 11, 23, 5, 17, 29, 31
24.	10	spc_char + upp_char	12, 23, 8, 19, 30, 32
25.	9	low_char + num_char + upp_char	3, 14, 9, 20, 5, 18, 25, 33
26.	7	num_char + upp_char	9, 20, 5, 17, 28, 32
27.	10	low_char + num_char + spc_char + upp_char	1, 13, 9, 20, 11, 22, 5, 18, 27, 34
28.	15	low_char + num_char + spc_char + upp_char	4, 16, 10, 21, 12, 23, 6, 19, 26, 33
29.	8	num_char + spc_char + upp_char	9, 20, 11, 22, 6, 18, 29, 32
30.	13	low_char + num_char + upp_char	3, 15, 10, 21, 7, 19, 25, 33

4. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Скільки чисел виведе цей цикл ?

```
for n in range(10, -1, -1):
    print(n)
```

2. Напишіть цикл, який обчислює суму всіх цілих чисел від 1 до *n* ?
3. Що виведе цей вкладений цикл ?

```
for i in range(3) :
    for j in range(1, 4) :
        print(i + j, end = "")
    print()
```

4. Напишіть цикл, який виводить:

   а. Всіх квадрати чисел менші ніж *n*. Для прикладу, якщо *n* рівне 100,

виведуться 0 1 4 9 16 25 36 49 64 81.

b. Всі цілі числа, кратні 10. та менші *n*. Для прикладу, якщо *n* рівне 100, виведуться 10 20 30 40 50 60 70 80 90

c. Всі степені двійки менші ніж *n*. Для прикладу, якщо *n* рівне 100, виведуться 1 2 4 8 16 32 64.

5. Напишіть цикл, який обчислює:
   a. Суму всіх парних чисел між 2 та 100 (включно).
   b. Суму всіх квадратів чисел між 1 та 100 (включно).
   c. Суму всіх непарних чисел між *a* та *b* (включно).
   d. Суму всіх непарних цифр числа *n*. (Для прикладу, якщо *n* рівне 32677, сума буде 3 + 7 + 7 = 17)
6. Скільки ітерацій виконає наступний цикл:
   a. for i in range(1, 11) . . .
   b. for i in range(10) . . .
   c. for i in range(10, 0, −1) . . .
   d. for i in range(−10, 11) . . .
   e. for i in range(10, 0) . . .
   f. for i in range(−10, 11, 2) . . .
   g. for i in range(−10, 11, 3) . . .
7. Перепишіть цикл з *for* на *while*:

```
s = 0
for i in range(1, 10) :
    s = s + i
```

8. Напишіть програму, яка приймає рядок та виводить:
   a. Лише літери у верхньому регістрі присутні в рядку.
   b. Кожну другу літеру в рядку.
   c. Кількість цифр в рядку.

5. СПИСОК ЛІТЕРАТУРИ

1. Learn to Program with Python 3. A Step-by-Step Guide to Programming, Second Edition / Irv Kalb. – Mountain View: Apress, 2018. – 361 p.
2. The Python Workbook. A Brief Introduction with Exercises and Solutions, Second Edition / Ben Stephenson. – Cham: Springer, 2014. – 218 p.
3. Python Pocket Reference, Fifth Edition / Mark Lutz. – Sebastopol: O'Reilly Media, Inc., 2014. – 264 p.
4. Learn Python 3 the Hard Way / Zed A. Shaw. – Boston: Addison-Wesley, 2017. – 321 p.
5. A Python Book: Beginning Python, Advanced Python, and Python Exercises / Dave Kuhlman. – Boston: MIT, 2013. – 278 p.

НАВЧАЛЬНЕ ВИДАННЯ

Написання програм з умовними виразами та циклами

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи № 4
з курсу «Програмування скриптовими мовами»
для студентів спеціальності
«Кібербезпека»

Укладач: Я. Р. Совин, канд. техн. наук, доцент