6. ОБРОБКА ТЕКСТІВ.

6.1.Рядки.

Рядки в мові Python є типом даних, спеціально призначеним для обробки текстової інформації. Рядок може містити текст довільної довжини, який обмежено наявною пам'яттю ПК.

У нових версіях Python є два типи рядків: звичайні рядки (послідовність байтів) і Unicode-рядка (послідовність символів). В Unicode-рядку кожен символ може займати в пам'яті 2 або 4 байти, в залежності від налаштувань періоду компіляції. Чотирьохбайтові знаки використовуються в основному для східних мов.

6.2.Кодування python-програми.

Для того, щоб Unicode-літерали в Python-програмі сприймалися інтерпретатором правильно, необхідно вказати кодування на початку програми, записавши в першому або другому рядку приблизно таке (для Unix/Linux):

 # -*- coding: koi8-r -*-
або (під Windows):
 # -*- coding: cp1251 -*-
Можуть бути й інші варіанти:
 # -*- coding: latin-1 -*-
 # -*- coding: utf-8 -*-
 # -*- coding: mac-cyrillic -*-
 # -*- coding: iso8859-5 -*-

Повний перелік кодувань (та їхніх псевдонімів):
import encodings.aliases
print(encodings.aliases.aliases)

{'646': 'ascii', 'ansi_x3.4_1968': 'ascii', 'ansi_x3_4_1968': 'ascii', 'ansi_x3.4_1986': 'ascii', 'cp367': 'ascii', 'csascii': 'ascii', 'ibm367': 'ascii', 'iso646_us': 'ascii', 'iso_646.irv_1991': 'ascii', 'iso_ir_6': 'ascii', 'us': 'ascii', 'us_ascii': 'ascii', 'base64': 'base64_codec', 'base_64': 'base64_codec', 'big5_tw': 'big5', 'csbig5': 'big5', 'big5_hkscs': 'big5hkscs', 'hkscs': 'big5hkscs', 'bz2': 'bz2_codec', …}

Якщо кодування не вказано, то вважається, що використовується us-ASCII, яке зберігає символи англійської мови, як числа що знаходяться між 0 та 127. (65 велике “A”, 97 маленьке “a”, і т.п.).

6.3.Рядкові літерали.

Python дуже багатий на операції з рядковими об'єктами. Рядки можна визначити

у програмі за допомогою рядкових літералів. Літерали записуються з використанням апострофів ', лапок " або цих самих символів, взятих тричі. Всередині літералів зворотна коса риска має спеціальне значення. Вона служить для введення спеціальних символів та для визначення символів через коди. Якщо перед рядковим літералом поставлено r, зворотна коса риска не має спеціального значення (r від англійського слова raw, рядок вказуєтья "як є"). Unicode-літерали вказуються із префіксом u. Наведемо кілька прикладів:

```
s1 = "рядок 1"
s2 = r'\1\2'
s3 = """apple\ntree"""# \n - символ переведення рядка
s4 = """apple
tree"""# рядок у потроєних лапках може мати всередині переведення рядків
s5 = '\x73\65'
u1 = u"Unicode literal"
u2 = u'\u0410\u0434\u0440\u0435\u0441'
```

Зворотна коса риска не повинна бути останнім символом у літералі, тобто, "str\" викличе синтаксичну помилку. Вказування кодування дозволяє використовувати в Unicode-літералах наведений на початку програми тип кодування. Якщо тип кодування не вказано, можна користуватися тільки кодами символів, визначеними через зворотну косу риску.

6.4.Операції над рядками.

До операцій над рядками, які мають спеціальну синтаксичну підтримку в мові, відносяться, зокрема конкатенація (склеювання) рядків, повторення рядка, форматування:

```
>>> print ("A" + "B", "A"*5, "%s" % "A")
AB AAAAA A
```

В операції форматування лівий операнд є рядком формату, а правий може бути або кортежем, або словником, або деяким значенням іншого типу:

```
print( "%i" % 234)
#234
print ("%i %s %3.2f" % (5, "ABC", 23.45678))
#5 ABC 23.46
```

a = 123
b = [1, 2, 3]
print("%(a)i: %(b)s" % vars())
#123: [1, 2, 3]

6.5.Модуль UNICOD.

Починаючи з версії 2.0, Python має новий тип даних для зберігання тексту: юнікодовий об'єкт. Він може використовуватися для зберігання та обробки даних у Unicode і добре інтегрується з існуючими рядковими об'єктами та здійснює автоматичну конверсію за потребою.

Перевага кодування Unicode полягає в тому, що воно визначає єдину ординальну величину для кожного символа у будь-якій письмовій системі сучасних чи давніх мов. До цього існувало лише 256 ординальних величин для позначення символів і текст здебільшого прив'язувався до кодування, що поєднувало ординальні величини з символами алфавіту. Це призводило до численних непорозумінь, особливо при інтернаціоналізації (цей термін традиційно позначається як "i18n" -- "i" + 18 символів посередині + "n") програмного забезпечення. Ця проблема вирішена в кодуванні Unicode, де одна кодова сторінка описує всі скрипти.

Створення юнікодових рядків у Python таке ж просте, як і створення звичайних рядків:

>>> u'Hello World !'
'Hello World !'

Маленький символ *"u"* перед лапками означає, що задається юнікодовий рядок. Задання спеціальних символів у рядку може бути зроблено за допомогою юнікодових контрольних послідовностей (Unicode-Escape encoding) мови Python, як це показано у наступному прикладі:

>>> u'Hello\u0020World !'
'Hello World !'

Контрольна послідовність *\u0020* вказує, що у заданій позиції повинен бути вставлений символ, що має ординальну величину *0x0020* (пробіл).

Інші символи інтерпретуються через пряме використання їхніх ординальних величин як ординальних величин кодування Unicode. Якщо ваші буквальні величини задано за допомогою кодування Latin-1, що використовується у багатьох західних

країнах, то для вас перші 256 символів кодування Unicode ті самі, що й 256 символів кодування Latin-1.

Окрім цих стандартних кодувань, Python має багато інших способів для створення юнікодових рядків на основі певного відомого кодування.

Вбудована функція unicode() надає доступ до всіх зареєстрованих юнікодових кодеків (codec < "COders and DECoders"). Серед найвідоміших кодувань, що можуть конвертуватися цими кодеками - Latin-1, ASCII, UTF-8, та UTF-16. Останні два — кодування змінної довжини, що зберігають юнікодові символи в одному чи більше байтах. Типове кодування — це здебільшого ASCII, що дозволяє лише симлоли від 0 до 127 і видає помилку, коли знаходить інші символи. Коли юнікодовий рядок виводиться на стандартний вивід, записується у файл чи конвертується за допомогою *str()*, то конверсія відбувається за допомогою цього стандартного кодування:

```
>>> u"abc"
'abc'
>>> str(u"abc")
'abc'
>>> u"&auml;&ouml;&uuml;"
'&auml;&ouml;&uuml;'
>>> str(u"&auml;&ouml;&uuml;")
'&auml;&ouml;&uuml;'
```

6.6.Методи рядків.

Таблиця 6.1 – методи рядків

Метод	Опис
center(w)	Центрує рядок у поле завдовжки *w*
count(sub)	Повертає кількість входжень рядка *sub* у рядок
encode([enc[, errors]])	Повертає рядок у кодуванні *enc*. Параметр *errors* може набувати значення "*strict*" (за замовчуванням), "*ignore*", "*replace*" або "*xmlcharrefreplace*"
endswith(suffix)	Чи закінчується рядок на *suffix*?
expandtabs([tabsize])	Заміняє символи табуляції на пробіли. За замовчуванням *tabsize*=8
find(sub [,start [,end]])	Повертає найменший індекс, із якого починається входження підрядка *sub* у рядок. Параметри *start* та *end* обмежують пошук вікном *start:end*, але повертається індекс, що відповідає вихідному рядку. Якщо підрядок не знайдено, повертається –1
index(sub[, start[,	Аналогічно **find()**, але генерує виняткову ситуацію **ValueError** у разі

end]])	невдачі
alnum()	Повертає *True*, якщо рядок містить тільки літери та цифри, і має ненульову довжину.Інакше – *False*
Isalpha()	Повертає *True*, якщо рядок містить тільки букви та має ненульову довжину
isdecimal()	Повертає *True*, якщо рядок містить тільки десяткові знаки (тільки для рядків Unicode) та має ненульову довжину
isdigit()	Повертає *True*, якщо містить тільки цифри та має ненульову довжину
islower()	Повертає *True*, якщо всі букви малі (і їх більше однієї), інакше – *False*
isnumeric()	Повертає *True*, якщо в рядку лише числові знаки (тільки для Unicode)
isspace()	Повертає *True*, якщо рядок складається тільки із символів пробілу. **Увага!** Для порожнього рядка повертається *False*
join(seq)	З'єднання рядків із послідовності *seq* через роздільник, указаний рядком
lower()	Робить усі літери в рядку малими
lstrip()	Видаляє символи пробілу ліворуч
replace(old, new[, n])	Повертає копію рядка, у якому підрядки *old* замінено на *new*. Якщо вказано параметр *n*, то заміняються тільки перші *n* входжень
rstrip()	Видаляє символи пробілу праворуч
split([sep[, n]])	Повертає список підрядків, що отримуються розбиттям рядка *a* роздільником *sep*. Параметр *n* визначає максимальну кількість розбиттів (ліворуч)
startswith(prefix)	Чи починається рядок із підрядка *prefix*?
strip()	Видаляє пробільні символи на початку й у кінці рядка
upper()	Робить усі літери в рядку великими

У наступному прикладі використовуються методи *split()* та *join()* для розбиття рядка у список (по роздільниках) та повторне об'єднання списку рядків у рядок:

```
s = "This is an example."
lst = s.split(" ")
print( lst)
#['This', 'is', 'an', 'example.']
s2 = "\n".join(lst)
print (s2)
#This
#is
#an
#example.
```

Щоб перевірити чи закінчується рядок певним сполученням літер, можна використати метод *endswith()*:

```
filenames = ["file.txt", "image.jpg", "str.txt"]
for fn in filenames:
    if fn.lower().endswith(".txt"):
        print(fn)
```

```
file.txt
str.txt
```

Шукати в рядку можна за методом *find()*. Наступна програма виведе перше входження символу *e* в рядку *st* починаючи із 7 позиції:

```
st = "Hello world. I'm a python developer"
print(st.find("e", 6))
```

```
27
```

Важливим для перетворення текстової інформації є метод *replace():*

```
>>> a = "Це текст , у якому є неправильно поставлені коми"
>>> b = a.replace(" ,", ",")
>>> print(b)
```

```
Це текст, у якому є неправильно поставлені коми
```

ПРАКТИЧНА РОБОТА

1) Використання рядків.

А)

```
s1 = "Рядок 1";s2 = 'Рядок 2'
print(s1, s2)
# формування рядка з іншого значення
s3 = str(8); print(s3)
# рядки, які складаються з багатьох рядків
s4 = """ Lesson2. Variables and Data Types
Some data types explained in this lesson:
- int, - bool, - float, - complex, - str """
print(s4)
# символ \ використовують, щоб продовжити рядок
# або будь-який вираз в Python з наступного рядка кода
s5 = "started\
continued"
print(s5)
```

Б)

```
string = "a string" # Створення рядка
# Виведення окремих символів рядка
print(string[0]) # 'a'
print(string[2]) # 's'
print(string[-1]) # 'g'
```

В)

```
string = "a string" # Створення рядка
# Виведення зрізів (групи символів) рядка
print(string[2:5]) # str
print(string[:5]) # a str
print(string[2:]) # string
print(string[::2]) # asrn
# Отримання окремих елементів рядка та їх конкатенація
print(string[2] + string[-3:]) # sing
```

Г)

```
string = input('Введіть рядок: ') # Введення рядка
```

```
# Перевірка, чи є в даному рядку символ «q»
if 'q' in string:
   print('В цьому рядку є символ "q"')
else:
   print('В цьому рядку немає символу "q"')
```

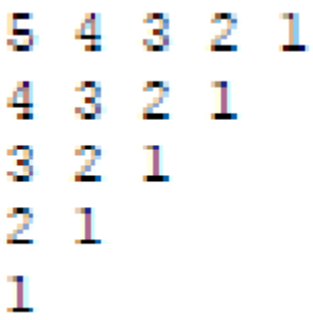
Д)

```
string = input('Введіть рядок: ') # Введення рядка
# Виведення довжини рядка
print(‘Довжина цього рядка:’, len(string))
```

2) Приклади операцій над рядками

```
str1 = 'hel'; str2 = 'lo'
result = str1 + str2 # конкатенація рядків
print(result)
# форматування рядків
a = 48;b = 73
message1 = '%d + %d = %d' %(a, b, a + b)
print(message1)
message2 = '{} - {} = {}'.format(a, b, a - b)
print(message2)
# індексація рядків
s = 'Hello, World!'
print(s[0]) # індексація розпочинається з нуля
print(s[4]) # четвертий (п‘ятий реально) елемент (символ)
print(s[-1]) # від‘ємні числа – індексація розпочинається з кінця
print(s[2:7])
# - символи з 2 (включно) по 7 (не включно)
print(s[2:7:2]) # теж саме, але з кроком два
```

3) Вводиться ціле число N (від 1 до 9), а виводяться рядки з числами, які утворюють визначений «рисунок»




```
try:
   N=int(input('Введіть N = '))
except Exception:
   print('Введіть число!!!')
else:
   M=N; pp=''
   while M!=0:
      i=M; L=[]
      while i!=0:
         if i<=M:
            L.append(str(i)); i-=1
      a=list(L)
      pp=''.join(a)
      print(pp); M=M-1
```

4) Визначити середнє арифметичне заданої непустої послідовності додатних цілих чисел, за якою слідує «0» (це ознака кінця послідовності).

```
from math import *
a=input ('Input first number: ')
if not a.isdigit():
   print("String value can not be entered")
   print("or number is negative, restart the program!")
   exit ()
else:
   a=int(a)
if a==0:
   print("The number can not be zero"); input ()

count=0;ar=0
while True:
   ar+=a; count+=1
   try:
```

```
        a=int(input('Input next number or Enter 0 to finish: '))
    except:
        print("String value can not be entered")
        print("or number is negative, restart the program!")
        exit ()
    else:
        if a==0: break

ar=ar/count; print("Average: ",ar)
```