Технически Университет – София,
Факултет „Компютърни системи и технологии“

Упражнение 3 – Оператори за разклонения и цикли. Разработване на програми с разклонена и циклична структура. Съставяне и настройка на програми с основните видове цикли върху потокови данни. Оператори за цикъл (while, for)

Условни оператори

Във всяка една програма се срещат условни оператори. Те позволяват да се изпълни една част от програмата(или да не се изпълни) в зависимост от стойността на логически израз. Логическите изрази могат да връщат само две стойности:True (истина) или False (лъжа), които съответстват на 1 и 0. Логическата стойност може да се пази в променлива:

```
x= True
y= False
print(x,y)
```

С логическа стойност True могат да се интерпретират обекти, чиято стойност е различна от нула или обект който не е празен.

Оператори за сравнение

В логическите изрази се използват следните оператори за сравнение:

== равно
!= различно
< по-малко
> по-голямо
<= по-малко или равно
>= по-голямо или равно
In проверява за съществуването на елемент в дадена последователност, връща True, ако елемента е намерен
Is проверява дали две променливи се отнасят към един и същи обект, ако е така връща True

Логически оператори

not обръща стойноста на логическия израз , ако е бил истина става лъжа, ако е лъжа става истина.

Логическо И and

Операторът and приема няколко условни израза и връща като резултат True или False. Логическото "И" връща True, само когато всички аргументи са със стойност True.

x	y	x and y
True	False	False
False	True	False
False	False	False
True	True	True

Логическо ИЛИ or

Логическо "ИЛИ" означава да е изпълнено поне едно измежду няколко условия, тогава резултата е True.

x	y	x or y
True	False	True
False	True	True
False	False	False
True	True	True

Пример : Каква е стойността на следните логически изрази ?

True или False ?

```
x=5
y=3
(x>8)or(y==2)      #False
(y==3) and(x<8)    #True
```

Оператор if – else

if – else е оператор за разклонение, който позволява в зависимост от стойността на логическия израз ако е True да се изпълни определен блок с код или да не се изпълни, ако стойността на логическия израз е False.

If<логически израз> :

< блок с код, който се изпълнява , ако логическия израз върне Истина>

else:

< блок с код , който се изпълнява , ако условието върне Лъжа>

Пример : Програма , която чете число ***n*** и проверява дали е по-голямо от 50, ако е така, тогава принтира съобщение ***n>50*** , в противен случай , принтира съобщение ***n<50***

Решение :

```
n = int(input('Enter number: '))
if n > 50:
    print('n>50')
else:
    print('n<50')
```

Оператор **if- else** позволява да бъдат поставени няколко условия с помощта на **elif**

Ето пример показващ неговата употреба : допълваме предишната задача , така че ако числото е равно на 50 да се принтира съобщение ***n=50***

```
n = int(input('Enter number: '))
if n > 50:
    print('n>50')
elif n < 50:
    print('n<50')
else:
    print('n=50')
```

Цикли

Циклите ни позволяват да се повтори изпълнението на даден блок с код , който се нарича тяло на цикъла.Броят на повторенията зависи от типа на цикъла, може да имате безкраен цикъл , който се изпълнява безкрайно и трябва да предвидите условие за изход от този тип цикли , за да избегнете зацикляне на програмата.

Цикъл for

Форматът на цикъл for е следния:

for< елемент > in <последователност >:
< тяло на цикъла >

Елемент е променлива, чрез която ще бъде достъпен текущия елемент при обхождането.

Последователност е обект, който може да се обхожда- низ, списък, кортеж, речник.
Тяло на цикъла съдържа операциите, които ще се изпълняват при всяка итерация на цикъла.

Пример : програма , която принтира числата от 1 до 10

```
for i in range(1, 11):
    print(i)
```

Цикълът започва с ключовата дума **for** , преминава през всички стойности за променливата **i** в дадения интервал от 1 до 11 ,без да включва 11 и за всяка стойност изпълнява **print(i)** т.е. отпечатва числото.

Пример : Да се напише програма, която отпечатва буквите от латинската азбука: a, b, …, z

```
for letter in range(ord('a'), ord('z')+1):
    print(chr(letter))
```

функция **ord('a')**-връща числовата стойност на подадения и като аргумент символ

функция **chr(letter)** – връща символа , който съответства на числената стойност подадена като аргумент на функцията
Цикъл **for** може да се използва и за обхождане на символите на една дума.
ord('a') връща числовата (Unicode/ASCII) стойност на символа 'a' (за 'a' това е 97).
ord('z') връща стойността за 'z' (122).

range(ord('a'), ord('z')+1) генерира числата от 97 до 122 включително — +1 е заради това, че range спира преди горната граница.

chr(letter) преобразува число обратно в символ.

for letter in ...: обхожда всички тези числови стойности;

print(chr(letter)) отпечатва съответната буква

Ето пример:

```
word = 'Python'
for letter in word:
    print(letter)
```

цикъл while

При този тип цикли , тялото на цикъла се изпълнява , докато е истина логическото условие.

while< логически израз>:

<тяло на цикъла >

Внимание : В тялото на цикъла трябва да е предвидено изменение на логическия израз, в противен случай рискувате да се получи зацикляне на програмата.

Пример : отпечатване на числата от 1 до 10

```
n = 1
while n < 11:
    print(n)
    n += 1
```

While True + break цикъл

Този тип цикъл повтаря фрагмент от кода многократно докато не се достигне до изрично прекратяване на цикъла, обикновено след if проверка в тялото на цикъла. Това е безкраен цикъл с проверка на дадено условие за изход вътре в тялото на цикъла. **while + break** изпълнява тялото си поне веднъж. В конструкцията на **while True + break** цикъла, условието винаги се проверява вътре в тялото му, докато при **while** цикъла проверката за изход от цикъла е винаги в началото, преди неговото тяло.

Пример: сумиране на цифрите на число

```
n = int(input('enter number: '))
sum = 0
while True:
    sum += n % 10     # sum = sum + n % 10
    n = n // 10
    if not n:         # if n == 0
        break
print(sum)
```

Оператори break и continue

Оператор break прекъсва изпълнението на цикъла и програмата продължава с изпълнение на първата команда след тялото на цикъла.
Оператор continue прекъсва текущата итерация на цикъла и осъществява преход към следващата итерация.

Пример : Програма, която принтира числата от 1 до 10, като пропуска числото 5

```
for number in range(1, 20):
    if number == 5:
        continue
    if number == 11:
        break
    print(number)
```

Функция **range()** – позволява да се генерира последователност с необходимата дължина.

range([начало] ,край [, стъпка])

Ако функцията има един параметър, това е параметър ***край***, в такъв случай за начало се приема 0. Параметър ***стъпка*** задава увеличител, по подразбиране стойността му е 1. Стъпката може и да е отрицателна.Стойността на параметъра ***край*** не се включва в създаваната последователност.

Пример : програма , която принтира числата от 20 до 10(не се включва) през 2.

```
for x in range(20, 10, -2):
    print(x, end=' ')
```

Изход : 20 18 16 14 12

Задачи:

1. Да се напише програма, която въвежда n цели числа (n > 0) и намира най-голямото между тях. Първо се въвежда броят на числата n. След това се въвеждат n цели числа

2. Да се напише програма, която въвежда n цели числа и ги сумира.

3. Да се напише програма, която въвежда число n и печата триъгълник от звездички

Примерен вход: **3**

Примерен изход: *****
******
****

4. Напишете програма, която проверява дали дадено число е просто, числото се въвежда от потребителя.