Жамбыл облысы әкімдігі білім басқармасы

«Білім» кәсіби гуманитарлық-техникалық колледжі

(білім беру ұйымының атауы)

**Оқу сабағының жоспары**

(теориялық немесе өндірістік оқыту)

While циклінда операторымен жұмыс барысы.

(сабақ тақырыбы)

**Модуль/Пән атауы** Бағдарламалық кодты қайта өңдеу

**Дайындаған педагог** Нургисаева У.М

**20\_**25**\_ жылғы** «\_\_\_\_» \_\_\_ақпан\_\_\_

**1. Жалпы мәліметтер**

Курс, оқу жылы, топ 3 курс, 3БҚ-22

Сабақ түрі: жаттығу

**2. Мақсаты, міндеттері:**

**Оқу:** Білім алушыларға python (пайтон) программалау тілінде While циклінда операторымен жұмыс ұйымдастыру, программаны өңдеу жұмыснегіздерін үйрету, шартты операторлардың жұмыс істеу принциптерін түсіндіру.

#### **Дамыту:** Логикалық ойлау қабілеттерін дамыту, есептердің әртүрлі шешімдерін табуға дағдыландыру, программалық кодтың құрылымын дұрыс құру қабілеттерін қалыптастыру.

#### **Тәрбиелік:** Жауапкершілік пен ұқыптылыққа тәрбиелеу, алгоритмдік ойлауды дамыту, шығармашылық қабілеттерін жетілдіру.

**3. Оқу-жаттығу процесінде білім алушылар меңгеретін күтілетін нәтижелер және кәсіби дағдылар тізбесі:** Python программалау тілінде While циклінда операторымен жұмыс ұйымдастыру, программаны өңдеу және пайдалану дағдыларын меңгеру. While циклі операторымен жұмыснегіздерін үйренеді, код жазады.

**4. Қажетті ресурстар:** ДК немесе ноутбуктер

[***https://stepik.org/lesson/597060/step/2?unit=592085***](https://stepik.org/lesson/597060/step/2?unit=592085)

***5. Сабақтың барысы: (90 минут)***

**5.1. Ұйымдастыру кезеңі:** *( 3 мин )*

**5.2. Үй жұмысын жан-жақты тексеру:**

***“Ойлан – Жұптас – Бөліс” әдісі*** *(15 минут)*

**While циклінда операторымен жұмыс барысы.**

Программалауда циклдер — белгілі бір шартқа негізделген қайталанатын операциялар жиынтығын орындауға мүмкіндік беретін құрылымдар. Циклдер арқылы программалаушылар көптеген операцияларды автоматты түрде қайталап, жүйелік ресурстарды тиімді пайдалана алады. Бүгінгі дәрісте біз *while* циклін және оның операторларымен жұмыс істеуді қарастырамыз.

***while* циклінің жалпы құрылымы**

*while* циклі — шарт тексеріліп, егер ол дұрыс (яғни, шын) болса, циклдің ішіндегі кодтың орындалуына мүмкіндік беретін конструкция. Ол әдетте келесідей құрылымда болады:

while шарт:

# цикл денесі

Бұл құрылымдағы *шарт* дегеніміз — белгілі бір логикалық тексеру. Егер шарт орындалса (яғни, шын болса), онда цикл денесінде жазылған операторлар орындалады. Цикл қайтадан шартты тексеруге оралады және егер шарт тағы да орындалса, цикл қайта орындалады. Бұл процесс шарт орындалмағанға дейін жалғасады.

while циклі тексерілетін шарт ақиқат болғанша бірдей әрекеттер тізбегін қайталап орындауға мүмкіндік береді. Шарт цикл денесінің алдында жазылады және цикл денесі орындалмас бұрын тексеріледі. Әдетте, уақытша цикл немесе циклдің орындалу санын нақты анықтау мүмкін болмаған кезде қолданылады.

while циклінің синтаксисі ең қарапайым жағдайда келесідей:

while шарт:

инструкция блогы

while циклін орындау кезінде алдымен шарт тексеріледі. Егер ол жалған болса, онда циклдің орындалуы тоқтатылады және басқару while циклінің денесінен кейінгі келесі командаға ауысады. Егер шарт ақиқат болса, онда оператор орындалады, содан кейін шарт қайтадан тексеріліп, оператор қайтадан орындалады. Бұл шарт дұрыс болғанша жалғасады. Шарт жалған болғаннан кейін цикл аяқталады және басқару циклден кейінгі келесі нұсқауға ауысады.

Мысалы, келесі бағдарлама фрагменті 1-ден 10-ға дейінгі барлық бүтін сандардың квадраттарын басып шығарады. Сіз төменде көрсетілген мысалдан while циклінің for … in range( … ) орнын алмастыра алатынын көре аласыз:

i = 1

while i <= 10:

print(i \*\* 2)

i += 1

Бұл мысалда цикл ішіндегі i айнымалысы 1-ден 10-ға дейін өзгереді. Циклдің әрбір жаңа өтуімен мәні өзгеретін мұндай айнымалы санауыш(счетчик) деп аталады. Осы фрагментті орындағаннан кейін i айнымалысының мәні 11-ге тең болатынын ескеріңіз, өйткені i == 11 кезінде i <= 10 шарты енді бірінші рет орындалмайтын болады.

Міне, n натурал санының цифрларының санын анықтау үшін while циклін пайдаланудың тағы бір мысалы:

n = int(input())

length = 0

while n > 0:

n //= 10 # бұл эквивалентті n = n // 10

length += 1

print(length)

Бұл циклде length айнымалысында мұның неше рет орындалғанын есептей отырып, сандардың соңынан бастап бір таңбалы сандарды алып тастаймыз, бұл бүтін санды 10-ға бөлуге тең (n //= 10).

Python-да бұл мәселені шешудің тағы бір жолы: length = len(str(n)).

# **Циклды басқару инструкциялары**

Цикл денесінен кейін else сөзін жазуға болады: одан кейін тексерілетін шарт жалған болған кезде цикл аяқталғаннан кейін бір рет орындалатын операциялар блогы:

i = 1

while i <= 10:

print(i)

i += 1

else:

print('Цикл аяқталды, i =', i)

Мұның мағынасы жоқ сияқты, өйткені дәл сол нұсқаулықты цикл аяқталғаннан кейін ғана жазуға болады. Мағынасы break инструкциясымен ғана бар болады. Орындау кезінде, егер Python цикл ішінде break инструкциясына тап болса, ол циклды орындауды дереу тоқтатады және одан шығады. Бұл жағдайда else тармағы орындалмайды. Әрине, break инструкциясы if операторының ішінде ғана шақырылуы керек, яғни ол қандай да бір ерекше шарт орындалған жағдайда ғана орындалуы керек.

Теріс санға тап болғанша сандарды оқитын бағдарламаның мысалы келтірілген. Теріс сан пайда болғанда, бағдарлама аяқталады. Бірінші нұсқада сандар тізбегі 0 санымен аяқталады (оны оқу кезінде тоқтату керек).

a = int(input())

while a != 0:

if a < 0:

print('Теріс сан кездесті', a)

break

a = int(input())

else:

print('Бірде-бір теріс сан кездескен жоқ')

Бағдарламаның екінші нұсқасында алдымен кіріске реттілік элементтерінің саны, содан кейін элементтердің өздері беріледі. Бұл жағдайда for циклін пайдалану ыңғайлы. for циклінде else те болуы мүмкін және оның ішінде break функциясы да болуы мүмкін.

n = int(input())

for i in range(n):

a = int(input())

if a < 0:

print('Теріс сан кездесті', a)

break

else:

print('Бірде-бір теріс сан кездескен жоқ')

Тағы бір циклды басқару инструкциясы – continue(циклдің жалғасуы). Егер бұл инструкция циклдің ортасында бір жерде кездессе, онда циклдың соңына дейін қалған барлық нұсқаулар өткізіп жіберіледі және циклдің орындалуы келесі итерациямен жалғасады.

Егер break және continue инструкциялары бірнеше кірістірілген циклдар ішінде болса, олар тек ішкі циклдің орындалуына әсер етеді. Міне, мұны көрсететін мысал:

for i in range(3):

for j in range(5):

if j > i:

break

print(i, j)

Мұнда break инструкциясын дұрыс пайдаланудың әдеттегі мысалы келтірілген (бұл код сандағы таңбалар санын есептейді).

n = int(input())

length = 0

while True:

length += 1

n //= 10

if n == 0:

break

print('Санның ұзындығы тең', length)

Бұл циклды келесідей қайта жазу әлдеқайда жақсы:

n = int(input())

length = 0

while n != 0:

length += 1

n //= 10

print('Санның ұзындығы тең', length)

Дегенмен, Python-да неғұрлым талғампаз шешім ұсынылуы мүмкін

n = int(input())

print('Санның ұзындығы тең=', len(str(n)))

**Циклдің жұмыс істеу принципі**

1. *while* циклі бірінші кезекте шартты тексереді.
2. Егер шарт орындалса, цикл денесіндегі код орындалады.
3. Код орындалған соң, шарт қайта тексеріледі.
4. Егер шарт әлі де орындалса, цикл қайта орындалады.
5. Шарт дұрыс болмаса, цикл тоқтайды және келесі операцияға өтеді.

***while* циклінің операторлары**

Цикл ішінде әртүрлі операторларды қолдануға болады. Олардың негізгі түрлері:

1. **Break операторы**  
   *Break* операторы циклді мәжбүрлеп аяқтайды. Яғни, егер белгілі бір шарт орындалса, циклды аяқтап, программаның келесі бөліміне өтуге мүмкіндік береді.

Мысалы:

count = 0

while count < 5:

print(count)

count += 1

if count == 3:

break

Бұл мысалда цикл 3-ке жеткенде тоқтайды, өйткені break операторы орындалады.

1. **Continue операторы**  
   *Continue* операторы циклдің ағымдағы итерациясын тоқтатып, шартты қайта тексеруге мүмкіндік береді. Бұл оператор ағымдағы қадамды өткізіп жіберуге мүмкіндік береді.

Мысалы:

count = 0

while count < 5:

count += 1

if count == 3:

continue

print(count)

Бұл мысалда 3-ке жеткенде, print(count) операциясы орындалмайды, бірақ цикл әрі қарай жалғасады.

1. **Else блогы**  
   *while* цикліне сәйкес келетін *else* блогы болады. Бұл блок цикл аяқталғаннан кейін орындалады, бірақ цикл *break* операторы арқылы ерте тоқтатылған жағдайда орындалмайды.

Мысалы:

count = 0

while count < 5:

print(count)

count += 1

else:

print("Цикл аяқталды")

Бұл жағдайда цикл толық орындалса, "Цикл аяқталды" деген жазба шығарылады.

**Циклдың тиімділігі мен қолдану аймақтары**

Циклдер программалауда жиі қолданылады, себебі олар көптеген бірдей әрекеттерді автоматты түрде орындауға мүмкіндік береді. *while* циклі әсіресе шарттар динамикалық түрде өзгеретін жағдайларда тиімді. Мысалы, пайдаланушыдан дұрыс енгізу алу немесе белгілі бір ресурстарды тексеріп отыру кезінде қолданылады.

### ****1-есеп: 1-ден N-ге дейінгі сандардың қосындысын есептеу****

N = int(input("N санын енгізіңіз: "))

summa = 0

i = 1

while i <= N:

summa += i

i += 1

print("1-ден", N, "дейінгі сандардың қосындысы:", summa)

### ****2-есеп: Санның цифрларын санау****

num = int(input("Санды енгізіңіз: "))

count = 0

while num > 0:

count += 1

num //= 10

print("Сандағы цифрлар саны:", count)

### ****3-есеп: Факториалды есептеу****

N = int(input("N санын енгізіңіз: "))

fact = 1

i = 1

while i <= N:

fact \*= i

i += 1

print(N, "факториалы:", fact)

### ****4-есеп: Берілген диапазондағы жұп сандарды шығару****

A = int(input("A санын енгізіңіз: "))

B = int(input("B санын енгізіңіз: "))

while A <= B:

if A % 2 == 0:

print(A, end=" ")

A += 1

|  |  |  |
| --- | --- | --- |
| **Білдім** | **Білемін** | **Білгім келеді** |
|  |  |  |

**Рефлексия**

**Үй тапсырмасы –** Практикалық жұмыс №4 орындау және қорғау