Циклы

while

```
saved_pwd = "right_password"

pwd = input("Введите пароль для входа: ")
while pwd != saved_pwd:
    pwd = input("Введите пароль для входа: ")

print("Пароль верный. Вход разрешён.")
```

- while, условие, двоеточие
- затем блок кода, который выполняется, пока условие возвращает True

Работает так:
1. Проверяется условие, если оно вернуло True - переход к шагу 2
2. Выполняется тело цикла, переход к шагу 1

Тело цикла — это код, который будет выполняться многократно

for i in range(n)

```python
n = int(input("Введите количество чисел: "))

for i in range(n):
    print(i, end=' ')

# Введите количество чисел: 5
# 0 1 2 3 4
```

```python
n = int(input("Введите количество чисел: "))

i = 0
while i < n:
    print(i, end=' ')
    i += 1

# Введите количество чисел: 5
# 0 1 2 3 4
```

- Сначала идёт for, итерируемая переменная, in, диапазон значений, двоеточие
- Затем идёт блок кода, который выполняется, пока итерируемая переменная шагает по диапазону

range

С помощью функции range() можно задавать диапазон значений для цикла for

Функция range() может принимать от одного до трёх целочисленных аргументов:

- range(n) – возвращает диапазон целых чисел от 0 до n - 1.
- range(k, n) – возвращает диапазон целых чисел от k до n - 1.
- range(k, n, s) – возвращает диапазон целых чисел от k до n - 1 с шагом s.

```
for i in range(4):
    print(i, end=' ')   # 0 1 2 3
print()

for i in range(1):
    print(i, end=' ')   # 0
print()

for i in range(1, 5):
    print(i, end=' ')   # 1 2 3 4
print()

for i in range(1, 2):
    print(i, end=' ')   # 1
print()
```

```
for i in range(1, 10, 2):
    print(i, end=' ')   # 1 3 5 7 9
print()

for i in range(9, -1, -2):
    print(i, end=' ')   # 9 7 5 3 1
print()

for i in range(0, 10, 1):
    print(i, end=' ')   # 0 1 2 3 4 5 6 7 8 9
print()

for i in range(9, -1, -1):
    print(i, end=' ')   # 9 8 7 6 5 4 3 2 1 0
print()
```

for example

Напишите программу, которая выводит все делители числа.

```
n = int(input('Введите число: '))

for i in range(1, n + 1):
    if n % i == 0:
        print(i, end=' ')
print()


# Введите число: 36
# 1 2 3 4 6 9 12 18 36
```

while example

Напишите программу, которая считает сумму вводимых чисел. Список завершается пустой строкой.

```
s = 0
x = input()

while x:
    s += int(x)
    x = input()

print('Сумма:', s)


# 10
# 22
# 32
#
# Сумма: 64
```

Вжух! А ещё вот так можно:

```
s = 0

while x := input():
    s += int(x)

print('Сумма:', s)
```

В Python 3.8 появился *моржовый оператор* (walrus operator).

Он записывается как ":=" и позволяет одновременно:

- вычислить выражение,
- присвоить результат переменной
- и вернуть это значение, например в условие.

Не путайте его с обычным присваиванием.

break

Циклы for и while можно останавливать при наступлении определённого условия. Для этого используется оператор break.

Можно после цикла поставить else. Оно выполнится если в цикле ни разу не сработал break.

```
password = "Виват, Лицей!"

while True:
    if input("Введите пароль: ") == password:
        print("Пароль принимается")
        break
```

```
# Введите пароль: Век живи, век Виват, Лицей!
# Введите пароль: Скажи мне, кто твой друг - и
я скажу Виват, Лицей!
# Введите пароль: Где родился - там и Виват,
Лицей!
# Введите пароль: Виват, Лицей!
# Пароль принимается
```

```
password = "Виват, Лицей!"

for i in range(3):
    if input("Введите пароль: ") == password:
        print("Пароль принимается")
        break
else:
    print('Попытки закончились')
```

```
# Введите пароль: 7 раз отмерь, один раз Виват, Лицей!
# Введите пароль: Лучше Виват, Лицей, чем журавль в небе
# Введите пароль: Сколько волка ни корми - всё равно
Виват, Лицей!
# Попытки закончились
```

continue

Оператор continue похож на break, но тогда как break полностью завершает цикл, continue завершает только текущую итерацию, а сам цикл продолжает выполняться.

```
a = 10

for i in range(3):
    b = int(input('Введите число: '))
    if b == 0:
        print('На ноль делить нельзя')
        continue
    if b == 1:
        print('На один делить неинтересно')
        continue
    print(f'{a} / {b} = {a / b :.2f}')


# Введите число: 0
# На ноль делить нельзя
# Введите число: 1
# На один делить неинтересно
# Введите число: 3
# 10 / 3 = 3.33
```

```
a = 10
i = 0
while i < 3:
    b = int(input('Введите число: '))
    if b == 0:
        print('На ноль делить нельзя')
        continue
    if b == 1:
        print('На один делить неинтересно')
        continue
    print(f'{a} / {b} = {a / b :.2f}')
    i += 1


# Введите число: 0
# На ноль делить нельзя
# Введите число: 1
# На один делить неинтересно
# Введите число: 3
# 10 / 3 = 3.33
# Введите число: 4
# 10 / 4 = 2.50
# Введите число: 5
# 10 / 5 = 2.00
```

Вложенные циклы

Внутри цикла `for` или `while` можно написать другой цикл.

Внешний цикл называется внешним, а тот, который внутри - внутренним.

На каждой итерации внешнего цикла будет один раз выполняться полный внутренний цикл.

```
n = 6
for i in range(1, n + 1):
    for j in range(1, n + 1):
        print(f'{i * j :2d}', end=' ')
    print()


#  1  2  3  4  5  6
#  2  4  6  8 10 12
#  3  6  9 12 15 18
#  4  8 12 16 20 24
#  5 10 15 20 25 30
#  6 12 18 24 30 36
```

Если во внутреннем цикле выполнится `break`, то текущий внутренний цикл завершится, но внешний продолжится.

```
n = 6
for i in range(1, n + 1):
    for j in range(1, n + 1):
        if i * j > 20:
            break

        print(f'{i * j :2d}', end=' ')
    print()

#  1  2  3  4  5  6
#  2  4  6  8 10 12
#  3  6  9 12 15 18
#  4  8 12 16 20
#  5 10 15 20
#  6 12 18
```

Вложенные циклы

Аня, Боря и Вова решили съесть апельсин, состоящий из n долек. Подскажите ребятам, как им его разделить. Напишите программу, которая выводит все возможные способы разделки апельсина.

```
n = int(input('Введите количество долек: '))

for i in range(1, n):
    for j in range(1, n):
        for k in range(1, n):
            if i + j + k == n:
                print(i, j, k)
```

```
n = int(input('Введите количество долек: '))

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

```
# Введите количество долек: 5
# 1 1 3
# 1 2 2
# 1 3 1
# 2 1 2
# 2 2 1
# 3 1 1
```

Вложенные циклы

Сложная кракозябра на подумать

```
n = 6
for i in range(1, n + 1):
    for j in range(1, n + 1):
        if i * j > 20:
            break
        print(f'{i * j :2d}', end=' ')

    else:
        print()
        continue

    break


#  1  2  3  4  5  6
#  2  4  6  8 10 12
#  3  6  9 12 15 18
#  4  8 12 16 20
```

Пример с предыдущего слайда для сравнения

```
n = 6
for i in range(1, n + 1):
    for j in range(1, n + 1):
        if i * j > 20:
            break
        print(f'{i * j :2d}', end=' ')

    print()



#  1  2  3  4  5  6
#  2  4  6  8 10 12
#  3  6  9 12 15 18
#  4  8 12 16 20
#  5 10 15 20
#  6 12 18
```

По строкам тоже можно итерироваться

В качестве диапазона можно поставить строку, итерируемая переменная будет шагать по её символам.

```
for e in 'aba caba':
    print(e, end=', ')   # a, b, a,  , c, a, b, a,
print()


for e in 'aba caba':
    if e in ' a':
        continue

    print(e, end=', ')   # b, c, b,
print()


for e in 'QsWuEmRmTaY UcIuOmP AlSaDuFdGeH':
    if e.isupper():
        continue

    print(e, end='') # summa cum laude
print()
```

По строкам тоже можно итерироваться

```
text = 'The first lesson at BSU Lyceum took place on January 10, 1990'
min_lower, max_lower = '', ''
min_upper, max_upper = '', ''

for e in text:
    if e.islower():
        if not min_lower:
            min_lower, max_lower = e, e
            continue

        min_lower = min(min_lower, e)
        max_lower = max(max_lower, e)
        continue

    if e.isupper():
        if not min_upper:
            min_upper, max_upper = e, e
            continue

        min_upper = min(min_upper, e)
        max_upper = max(max_upper, e)
        continue

print(min_lower, max_lower)     # a y
print(min_upper, max_upper)     # B U
```