Flow – потоки

Инструкции управления потоком:

Логические операторы управляют чаще всего данными в формате boolean:

* and
  + логика проверки:
    - 1 and 1 проверяет первый=True, проверяет второй операнд = True, возвращает значение последнего = 1
    - 0 and 1 проверяет первый=False, нет смысла проверять второй операнд, возвращает = 0
    - 1 and 0 проверяет первый= True, проверяет второй операнд = False, возвращает = 0
    - 2 and 3 проверяет первый=True, проверяет второй операнд = True, возвращает значение последнего =3
    - ‘ ‘ (пробел в кавычках) and 1 проверяет первый=True, проверяет второй операнд = True, возвращает значение последнего =1
    - ‘‘ (ничего) and 1 проверяет первый= False, нет смысла проверять второй операнд, возвращает ‘‘ (ничего)
* or
  + логика проверки:
    - False and False проверяет первый= False, проверяет второй операнд = False, возвращает = False (единственный случай)
    - 1 or 1 проверяет первый=True, нет смысла проверять второй операнд, возвращает = 1
    - True and False проверяет первый=True, нет смысла проверять второй операнд, возвращает = True
    - 0 or 1 проверяет первый=True, проверяет второй операнд, возвращает значение последнего = 1
* not – не возвращает значение, только Boolean (False или True), противоположный условию
  + not 1 = False
  + not 0 = True
  + not ‘‘ (ничего) = True

**IF**

if a:

do this

elif b:

do that

else:

whatever

| **Оператор відносин** | **Значення** |
| --- | --- |
| < | Менше |
| <= | Менше або дорівнює |
| > | Більше |
| >= | Більше або дорівнює |
| == | Дорівнює |
| != | Не дорівнює |

age = int(input('Enter your age for entering to our bar Duck: '))

message = age

if 21 <= age < 100:

    print('Welcome!')

elif age >= 100:

    print('Wow!!!')

elif 0 < age < 21:

    print('Too early for you')

else:

    print('Wrong input')

print('Thanks anyway')

Агоритм проверки возраста для доступа:

-если от 21 до 100 – вход

- если больше 100 вход+вау

- от 1 до 21 – запрет

- 0 неправильный ввод

пример тернарного выражения (в одну строку)

age = int(input('Enter your age for entering to our bar Duck: '))

message = 'Welcome' if 21 <= age < 100 else 'Wrong input'

print(message)

однако нельзя злоупотреблять такими выражениями в ущерб легкости чтения кода.

Операторы цикла

**цикл while –** когда не знаем сколько раз повторить

выполняет тело цикла до тех пор, пока соблюдается условие =True, когда условие перестает соблюдаться =False - выполняет следующую инструкцию

name = input('What is your name?: ')

while not name:

    name = input('What is your name?: ')

print('Welcome ' + name)

Запрос имени, пока не будет введено

В цикл while теоретически можно вложить бесконечное множество других циклов, но это будет противоречить принципам языка о простоте

* break – оператор прерывания любого цикла, которым можно настраивать нужные условия выхода из цикла (не совершив ни одного прохода цикла, после определенного кол-ва раз, и т.д.)

import random

while True:

    number = random.randint(0, 3)

    user\_number = int(input('Enter your number from 0 till 3: '))

    if number == user\_number:

        print('You won!!!')

        break

    else:

        print('Try again!')

print('Game over')

Игра – угадай номер, играть до победы, в случае победы – выход

shopping\_list = ''

while True:

    goods = input('What do you need more?: ')

    if goods == 'stop':

        print('You list is: ' + shopping\_list)

        break

    shopping\_list += goods

Составление списка покупок путем ввода дополнения в переменную и вывод в конце ее значения

**цикл FOR LOOP –** когда знаем сколько раз повторить

for I in iterative:

pass

range(start, stop, step)

используется для того чтобы повторить цикл определенное количество раз или перебрать какой-то итерируемый объект

«in» - проверка входит ли элемент в последовательность или нет

например:

‘a’ in ‘abc’ = True

‘a’ not in ‘abc’ = False

‘d’ not in ‘abc’ = True

‘pass’ – заглушка, которая ничего не делает, можно использовать во многих местах. Например при разработке структуры, когда основное действие еще не придумано, но во избежание синтаксических ошибок временно используется заглушка.

a = "abcdefgijhk"

for letter in a:

    print(letter)

    if letter == "g":

        break

a = "abcdefgijhk"

i = 0

for letter in a:

       i += 1

       print(i)

    print(letter)

print(letter) выведет по очереди каждый знак в строке,

если print(i) выведет по очереди цифры от 1 до …(кол-во знаков в строке)

нельзя итерировать число в формате int (использовать 123 вместо abcdefgijhk в А ), но часто нужно именно итерировать именно числовое значение.

Для этого есть функция

range(start, stop, step)

range(0,10,1) шаг 1 по умолчанию, поэтому опускается. сгенерирует от 0 до 9 (не включает последний аргумент)

for letter in range(0, 12):

    print(letter)

выведет 12 чисел от 0 до 11

i – принято использовать при работе с числами, а внятное имя с прочими (если есть вложенный в него цикл, то вводят следующую переменную – j, глубже не нужно – меняем логику или операторы), поэтому правильнее будет:

for i in range(0, 12):

    print(i)

start = int(input("Start: "))

stop = int(input("Stop: "))

for i in range(start, stop+1):

    print(i)

выполнит вывод чисел от заданного до заданного

циклы можно комбинировать и вкладывать в любых вариациях

имя «range» логично использовать в числовых значениях, при переборе прочих значений нужно задавать внятное имя

***Ошибки, исключения***

Базовые (встроенные) или кастомные (созданные самостоятельно)

Базовые (очень много) делятся прерывание с помощью клавиатуры и исключения в целом:

- атрибутов

-арифметики

-имен

-циклов

-и тд

Принципы:

– чем раньше остановить код при ошибке и уведомите пользователя – тем лучше

- не игнорировать ошибки и не прятать их

Обработка ошибок:

try:

int(‘abc’)

except ValueError:

print(‘WTF?’)

except(ValueError, TypeError): вариант для нескольких исключений с единым решением проблемы

else:

print(‘There were no problem’) – выполнится только если код прошел нормально, если ошибка – то блок не пройдет

finally:

print(‘Anyway’) – выполнится всегда. нужен для очистки, сохранения данных, освобождения портов и т.д