**Print**

Print(… , … , **sep=**) sep= - через шо в нас буде все виводитись. НП sep=”, ” (через ‘ ’ та ‘,’)

Print(… , … , **end=**) end= - кінцевий символ. НП end=”.” (в кінці буде ставитись ‘.’)

Print(“**\n**”) \n – перехід на новий рядок (писати в “”)

Print(“**\t**”) \t – табуляція (теж в “”)

Print(“**\(щось)**”) \ - для виводу якихось символів. НП print(“john**\’**s”) (щоб вивело ‘ )

Print(“**\\**”) \\ - вивести \

**Округлити число**

pi = 3.1415926535

print(f'{pi:.3f }') – округли до 3 знаків

print(f'{pi:.1f }') – до одного

**Вставка запятых в качестве разделителя**

number = 1234567890.12345

print (f ’ {number:,}’)

1,234,567,890.12345

///

number = 1234567890.12345

print (f' {number:,.2f}')

1,234,567,890.12

**Форматирование числа с плавающей точкой в процентах**

discount = 0.

print (f' {discount:%} ')

50.000000%

///

print (f ' { discount :.0%}') - 0 знаків після коми

50%

**Форматирование в научной нотации**

number = 12345.6789

print( f ‘ {number :e}')

1.234568e+04

print ( f ' {number: .2e)')

1.23e+0

**Форматирование целых чисел**

Print(f’{num:d}’) – ставиться буква d

Print(f’{num:,d}’) – щоб виводити через кому

І тд

**Указание минимальной ширины поля**

number = 12345.6789

print(f'Число равняется {number:12,.2f}') – важливо вказувати ширину поля першу

Число равняется 12,345.68

**Выравнивание значений**

Условное обозначение выравнивания

< Выровнять значение по левому краю

> Выровнять значение по правому краю

^ Выровнять значение по центр

Print(f'{number:<10d}')

Print(f'{number:>10d}')

Print(f'{number:^10d}')

**Порядок следования условных обозначений**

*[выравнивание] [ширина] [,] [ . точность] [ тип]*

**Turtle**

Import turtle

Turtle.showturtle() – показати курсор

turtle.hideturtle() – ховає курсор, АЛЕ не змінює те як малює черепаха

turtle.forward(n) – перемістити вперед на n пікселів

turtle.right (90)/ turtle.left(90) – повертає курсор в вказаний напрямок на вказаний градус

turtle.setheading (90) – ставить курсор на вказаний градус

turtle.heading() -виводить який зараз градус у курсора

turtle.penup() – підняти перо

turtle.pendown() – опустити перо

turtle.circle(100) – щоб намалювати круг з вказаним радіусом

turtle.dot() – малює точку

turtle.pencsize(5) – вказуєм ширену пера (в пікселях), (якщо нічого не вказано повертає який зараз розмір пера)

turtle.pencolor ('red') – вказуєм колір пера (Найпоширеніші: 'red', 'green', 'blue', 'yellow' и 'cyan') (Якщо нічого не вказано, повертає який зараз колір)

turtle.bgcolor ('gray') – ставимо колір фона, ) (Якщо нічого не вказано, повертає який зараз колір)

**Возвращение экрана в исходное состояние**

turtle.reset () - стирает все рисунки, которые в настоящее время видны в графическом окне, задает черный цвет рисунка и возвращает черепаху в ее исходное положение в центре экрана. **Эта команда не переустанавливает цвет фона графического окна.**

turtle.clear () - просто стирает все рисунки, которые в настоящее время видны в графическом окне. **Она не меняет положение черепахи, цвет пера или цвет фона графического окна.**

turtle.clearscreen() - стирает все рисунки, которые в настоящее время видны в графическом окне, переустанавливает цвет пера в черный, переустанавливает цвет фона графического окна в белый и возвращает черепаху в ее исходное положение в центре графического окна

**///**

turtle.setup(640, 480) – встановлюєм ширишу і висоту вікна

turtle.goto (0, 100) – вказуєм куди перемістити черепаху

turtle.pos() – текущая позиция курсора

turtle.xcor ()/turtle.ycor () – для відображення х або у кординат

turtle.speed (скорость) – встановлюєм скорость курсора, 0 – мнговено, від 1 до 10 буде 1-сама повільна, 10- сама швидка (якщо не вказувати аргумент то можна отримати текещую скорость курсора)

turtle.write('Привет, мир!') – виводить текст в вікно

**Заполнение геометрических фигур**

import turtle

turtle.hideturtle ()

turtle.fillcolor ('red') – встановлює колір заливки, ) (Якщо нічого не вказано, повертає який зараз колір)

turtle.begin\_fill ()

turtle.circle(100) для заливки фігури

turtle.end\_fill ()///

**Пучение входных данных с помощью диалогового окна**

переменная = turtle.numinput(заголовок, подсказка) (якщо замість ОК пользователь нажме CANCEL то функція поверне None)

переменная = turtle.numinput(заголовок, подсказка, default=x, minval=y, maxval=z)

♦ Аргумент default=x указывает на то, что вы хотите, чтобы значение х изначально выводилось в поле ввода. Это позволяет пользователю просто нажимать кнопку ОК и принимать значение по умолчанию.

♦ Аргумент minval=y указывает на то, что вы хотите отклонять любое вводимое пользователем число, которое меньше у. Если пользователь введет число меньше у, то появится сообщение об ошибке и диалоговое окно останется открытым.

♦ Аргумент maxval=z указывает на то, что вы хотите отклонять любое вводимое пользователем число, которое больше z. Если пользователь введет число больше z, то появится сообщение об ошибке и диалоговое окно останется открытым.

Вот пример инструкции, в которой используются все необязательные аргументы:

num = turtle.numinput('Требуются данные', 'Введите значение в интервале 1-10’,default=5, minval=1, maxval=10)

**Получение строковых входных данных с помощью команды turtle.textinput**

переменная = turtle.textinput(заголовок, подсказка)

**Использование turtle.done() для оставления графического окна открытым**

Пишемо turtle.done() в кінці програми щоб вікно саме не закривалось по завершенню програми

///

turtle.heading() – визначає під яким кутом черепаха

turtle.isdown() – перенвіряє чи перо опущене

turtle.isvisible() – перевіряєм чи курсор видно чи ні

**Math**

\*\* - степіть. НП 2 \*\* 3 (буде 8)

Min(… , …, …, …) – знаходить мінімальне з переліку

Max() – так само як min

Abs() – модуль

Pow() – степінь

Round() – округляє число. НП round(6.7) буде 7

a+= - додає число до чогось (-=, /=, \*=, %=)

**Input**

Input() – ввести щось з консолі. НП input(“Enter number: “)

!Важливо: input() завжди приймає тип даних **string**

Int(input()) – приймає і робить int. НП a = int(input())

Float(input())

String(input())

Boll(input())

**Змінні і типи даних**

Del (якась зміна) – Видаляє змінну. НП del number

None – тип даних де немає нічого (НП acd=None)

**Умовні конструкції**

If (умова):

…

Elif (умова):

…

Else:

…

If not a:

Print(“not a”)

And/or – і/або

Скорочений варіант:

Data=”info”

Correct = true if data == “info” else false

**Цикли**

**For** i **in range(1,11,2):**

**…**

(1,11,2) – з 1 починається, до 11(з 0 починаєм), з кроком 2

////

Word=”some text”

For I in word:

Print(i)

Ми будем проходитись по тексту

///////

I=0

While i==10:

Print(i)

I+=1

////

Break – виходить з цикла

Continue – пропускає якесь коло в циклі

Range(10,0,-1) – від 10 до 1

////

For I in “hello world”:

If I == ‘v’:

Break

Else:

Print(“not found”)

**Короткі цикли**

**[вьражение\_результата выражение\_итерации]**

List1 = [1, 2, 3, 4]

list2 = [item for item in list1] – копіювання списка 1 в список 2

len\_list=[len(s) for s in str\_list]

**+if**

**[вьражение\_результата выражение\_итерации ycловие\_if]**

List1= [1, 12, 2, 20, 3, 15, 4]

list2= [item for item in list1 if item < 10]

**In**

Значення in список – повертає True or False

1 in [2,3,6,8,10,1] – верне True

**If \_\_name\_\_ == ‘\_\_main\_\_’**

If \_\_name\_\_ == ‘\_\_main\_\_’:

Main()

Else:

….

(! If буде спрацьовувати якщо це ми запускаєм як основну програму, якщо цей файл ми використовуєм як бібліотеку то буде спрацьовувати else)(Коротше кажучи якшо ми компілюєм цей файл то працює if, якщо ми компілює інший файл де цей файл використовується як бібліотека то спрацює else)

**Списки**

Mas=[5, ’4’, 5.56, true, “3”] (можна вводити що хочеш)

Print(mas[1:3:2])

З **По ідексу** до **По факту** **З кроком**

**Оператор повторення**

1 » numbers = [0] \* 5

2 » print (numbers)

3 >>> [0 , 0 , 0 , 0 , 0 ]

///

1 » numbers = [1, 2, 3] \* 3

2 » print(numbers)

3>>> [1, 2 , 3 , 1 , 2 , 3 , 1, 2 , 3 ]

**Зєднати два списка**

List1 = [1, 2, 3, 4]

list2 = [5, 6, 7, 8]

list3 = listl + list2 або list1 += list2

Print(mas[2:]) – Будем виводити всі елементи починаючи з 2 (**по індексу**)

Print(mas[2:-1]) - Будем виводити всі елементи починаючи з 2 (**по індексу**) але до останнього

Print(mas[:-1]) - Будем виводити всі елементи до останнього

Print(mas[:3]) – буде виводити елементи до 3

Print(mas[::2]) – виводим всі елементи з кроком два( тобто через один)

Print(mas[::-1]) – виводим всі елементи у зворотньому порядку

Якщо ми працюєм з відємним кроком print(mas[:3:-1]) то всі знчення мають бути відємні print(mas[:-3:-1])

////

Mas =[1,2,3,[4,5,6]]

Print(mas[3][1]) (виведе 5)

/////

Mas = [1, 2, 3]

Print(mas[-1]) (Виведе останій елемент, якщо написати -2 виведе передостаній, і тд)

/////

**Пройтись по двовимірному масиву**

for row in values:

for element in row:

print(element)

///

**Функції масива**

Mas.append() – добавляє в кінець списку елемент

Mas.insert(1, ”wow”) **– (Вставляє значення у задану індексну позицію у списку**. Коли значення вставляється у список, список розширюється, щоб розмістити нове значення. Значення, яке раніше знаходилося в заданій індексній позиції, і всі елементи після нього зсуваються на одну позицію до кінця списку. Якщо вказано неприпустимий індекс, виняток не відбувається. Якщо заданий індекс за межами кінця списку, значення буде додано **до кінця список**. Якщо застосовано негативний індекс, який вказує на неприпустиму позицію значення буде вставлено на **початок списку)**

Mas.extend([1,2,3,4]) – додає до списку інший список

Mas.sort() – сортує спикос в порядку зростання (!! Список має бути одного типу даних) (str також сотрує)

Mas.reverse() – виводить список задом на перед

Mas.pop() – видаляє останій елемент списку

Mas.remove(5) – Видаляє зі списку першу появу значення. Якщо значення у списку не знайдено, викликається виключення ValueError

Mas.clear() – очищає список

Mas.count(2) – шукає скільки в нас є елементів з вказаним значеням (НП 2)

Len(…) – вказує довжину (НП довжину списку)

Mas.index() – Повертає індекс першого елемента, значення якого дорівнює значенню. Якщо значення у списку не знайдено, повертає ValueError

**Строковые методы**

**isalnum()** - Возвращает истину, если строковое значение содержит только буквы алфавита или цифры и имеет по крайней мере один символ. В противном случае возвращает ложь

**isalpha()** - Возвращает истину, если строковое значение содержит только буквы алфавита и имеет по крайней мере один символ. В противном случае возвращает ложь

**isdigit()** - Возвращает истину, если строковое значение содержит только цифры и имеет по крайней мере один символ. В противном случае возвращает ложь

**islower()** - Возвращает истину, если все буквы алфавита в строковом значении находятся в нижнем регистре, и строковая последовательность содержит по крайней мере одну букву алфавита. В противном случае возвращает ложь

**isspace()** - Возвращает истину, если строковое значение содержит только пробельные символы и имеет по крайней мере один символ. В противном случае возвращает ложь. (Пробельными символами являются пробелы, символы новой строки (\п ) и символы табуляции (\t).)

**isupper()** - Возвращает истину, если все буквы алфавита в строковом значении находятся в верхнем регистре, и строковая последовательность содержит по крайней мере одну букву алфавита. В противном случае возвращает ложь

|  |  |
| --- | --- |
| **Lower()** | Возвращает копию строкового значения, в котором все буквы преобразованы в нижний регистр. Любой символ, который уже находится в нижнем регистре или не является буквой алфавита, остается без изменения |
| **Istrip()** | Возвращает копию строкового значения, в котором все ведущие пробельные символы удалены. Ведущими пробельными символами являются пробелы, символы новой строки (\п) и символы табуляции (\t), которые появляются в начале строкового значения |
| **lstrip(символ)** | Аргументом символ является строковое значение, содержащее символ. Возвращает копию строкового значения, в котором удалены все экземпляры символа, появляющиеся в начале строкового значения |
| **rstrip()** | Возвращает копию строкового значения, в котором все замыкающие пробельные символы удалены. Замыкающими пробельными символами являются пробелы, символы новой строки (\п) и символы табуляции (\t), которые появляются в конце строкового значения |
| **rstrip(символ)** | Аргументом символ является строковое значение, содержащее символ. Возвращает копию строковой последовательности, в которой удалены все экземпляры символа, появляющиеся в конце строкового значения |
| **Strip()** | Возвращает копию строкового значения, в котором удалены все ведущие и замыкающие пробельные символы |
| **strip(символ)** | Возвращает копию строкового значения, в котором удалены все экземпляры символа, появляющиеся в начале и конце строкового значения |
| **upper()** | Возвращает копию строкового значения, в котором все буквы преобразованы в верхний регистр. Любой символ, который уже находится в верхнем регистре или не является буквой алфавита, остается без изменения |

|  |  |
| --- | --- |
| **endswith(подстрока)** | Аргумент подстрока — это строковое значение. Метод возвращает истину, если строковое значение заканчивается подстрокой |
| **find(подстрока)** | Аргумент подстрока — это строковое значение. Метод возвращает наименьший индекс в строковом значении, где найдена подстрока. Если подстрока не найдена, метод возвращает -1 |
| **replace(старое,новое)** | Аргументы старое и новое — это строковые значения. Метод возвращает копию строкового значения, в котором все экземпляры старых подстрок заменены новыми подстроками |
| **startswith(подстрока)** | Аргумент подстрока — это строковое значение. Метод возвращает истину, если строковое значение начинается с подстроки |

копируемое\_строковое\_значение \* n – повторити строкове значення n раз

wordlist = my\_string.split() – розбиває строку і додає все в список (за замовченям розділювач стоїть “ ” – пробіл, но можна вказати свій в дужках через ‘’)

my\_string = 'Один два три четыре’

['Один', 'два', 'три', 'четыре']

**Функції**

Type() – визначати який тип даних

.join() – поєднує НП список до чогось (НП word=””.join(text) )//(word=”, ”.join(text)(Кожна буква буде писатись через “, ”))

.capitalize() – перші символи в слові з великої а решта з малої

.ceil() – округляє вказане число (import math)

.replace(“1”,”2”) – замінює перше значення на друге

.count() – рахує скільки раз входив символ в строку

**Tuple/Кортежі**

Їхні значення не можна змінювати як в list але вони менше важать

Data = (5,6,-1,”wow”, True, [1,2,3], (3,4,5) )

Tuple() – робить з чогось кортеж

///

Num=[1,2,3]

Data = tuple(num)

My\_tuple = (1,) – щоб створити кортеж з одним елементом

My\_tuple = (1) – просто створить змінну my\_tuple а не кортеж

Кортежи не поддерживают методы append (), remove (), insert (), reverse () и sort ().

**Dict/Словник**

Per = {‘name’ : ’max’, ‘age’ : 15, 5 : 11, True: “False”, 5.43: “WOOW”, (1,2): 23}

**Ключ/індекс ндаємо значення**

Print(per[‘age’])

Інший спосіб записання

Per1=dict(name=”wow”,age=15)

**Ключ** **значення**

Print(per1[**‘**age**’**])

**Лапки потрібні**

//// Вивести словник через for

For key, val in per.items():

Print(key, val, sep=”--- ”)

/////

For I in per.values():

Print(i)

////

Функції словника

**Додати новий елемент**: per[‘key’]= ‘value’

имя\_словаря[ключ] = значение

Здесь имя\_словаря— это переменная, которая ссылается на словарь, ключ— это применяемый ключ. Если ключ уже в словаре существует, то присвоенное ему значение будет заменено значением. Если же ключ отсутствует, то он будет добавлен в словарь вместе со связанным с ним значением.

del имя\_словаря [ ключ ] – щоб видалити якись ключ і його значення

my\_dict = {} or my\_dict = dict() – **створюєм пустий словник**

|  |  |
| --- | --- |
| **.clear()** | Очищает содержимое словаря |
| **.get()** | Получает значение, связанное с заданным ключом. Если ключ не найден, этот метод не вызывает исключение. Вместо этого он возвращает значение по умолчанию |
| **.items()** | Возвращает все ключи в словаре и связанные с ними значения в виде последовательности кортежей |
| **.keys()** | Возвращает все ключи в словаре в виде последовательности кортежей |
| **.pop()** | Возвращает из словаря значение, связанное с заданным ключом и удаляет эту пару "ключ : значение". Если ключ не найден, этот метод возвращает значение по умолчанию |
| **.popitem()** | Возвращает в виде кортежа последнюю добавленную в словарь пару "ключ : значение". Этот метод также удаляет пару "ключ : значение" из словаря |
| **.values()** | Возвращает все значения из словаря в виде последовательности кортежей |

**словарь.get(ключ, значение\_по\_умолчанию)** - В данном формате словарь — это имя словаря, ключ— это искомый в словаре ключ, значение\_по\_умолчанию— значение, которое возвращается, если ключ не найден. Когда этот метод вызывается, он возвращает значение, которое связано с заданным ключом. Если заданный ключ в словаре не найден, то этот метод возвращает значение\_по\_умолчанию.

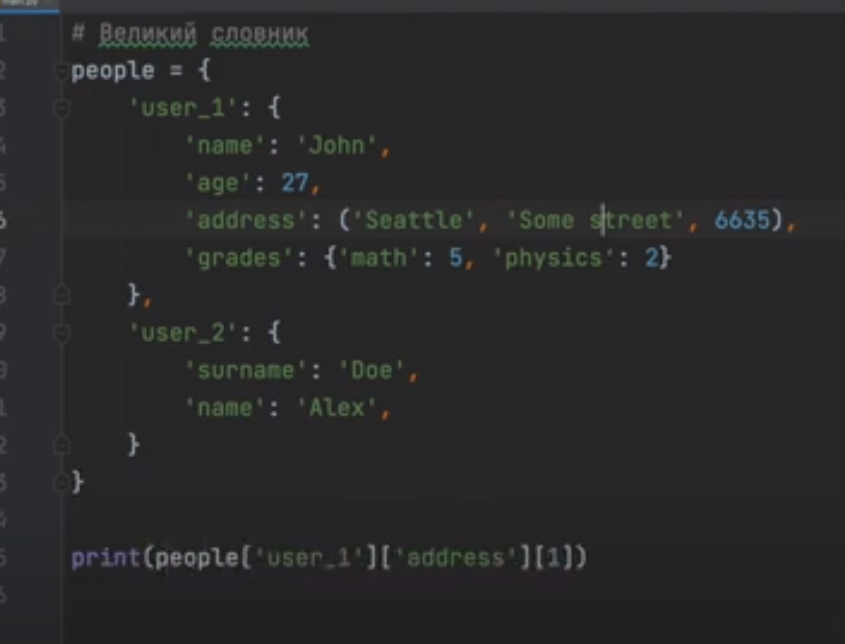
**словарь.pop(ключ, значение\_по\_умолчанию)** - В данном формате словарь — это имя словаря, ключ— это искомый в словаре ключ, значение\_по\_умолчанию— значение, которое возвращается, если ключ не найден. Когда этот метод вызывается, он возвращает значение, которое связано с заданным ключом, и удаляет эту пару "ключ : значение" из словаря. Если заданный ключ в словаре не найден, то метод возвращает значение\_по\_умолчанию

**Метод .popitem()** - выполняет два действия: во-первых, удаляет пару "ключ : значение", которая была в последний раз добавлена в словарь, и во-вторых, возвращает эту пару "ключ : значение" в виде кортежа. Вот общий формат метода:

словарь.popitem()

к, v = словарь.popitem()

Следует иметь в виду, что если метод popi tem () вызывается с пустым словарем, то он вызывает исключение KeyError.



**Включення в словник**

{1:1, 2:4, 3:9, 4:16} – ми хочемо зробити це

squares = {item:item\*\*2 for item in numbers}

phonebook = {'Крис':'555-1111', 'Кэти':'555-2222', 'Джоанна':'555-3333'}

phonebookcopy = {k:v for k,v in phonebook.items()}

populations = {'Нью-Йорк': 8398748, 'Лос-Анджелес': 3990456, 'Чикаго' : 2705994, 'Хьюстон' : 2325502, 'Феникс' : 1660272, 'Филадельфия' : 1584138} – хочемо створити словник з населеням більше 2 000 000

largest = {k:v for k,v in populations.items () if v > 2000000}

**if с операциями включения в словарь**

{выражение\_результата выражение\_итерации условие\_if}

largest = {k:v for k,v in populations.items () if v > 2000000}

**Множини/set**

Вони завжди перемішуються і в нихи не можуть повторятись елементи

Data={1, 2, 4.5, True, “wow”}

myset = set('ааабв')

Символ 'а' встречается в строковом значении многократно, но в множестве он появится

только один раз. После исполнения этой инструкции переменная myset будет ссылаться на

множество, содержащее элементы 'а', 'б' и 'в'.

Функції

Len()

.pop() – видаляє останій елемент

.clear() – очищає множину

.add() – додати якись елемент

.update([… , … , ….]) – додати декілька елементів

При вызове метода updated в качестве аргумента передается объект, который содержит итерируемые элементы, такие как список, кортеж, строковое значение или другое множество. Отдельные элементы объекта, передаваемого в качестве аргумента, становятся элементами множества. Приведенный ниже интерактивный сеанс это демонстрирует:

1 » > myset = set([l, 2, 3])

2 » > myset.update ( [4, 5, 6])

.remove() – видаляє елемент з вказаним значенням (Якщо елемента нема визиває помилку)

.discard() - видаляє елемент з вказаним значенням (Якщо елемента нема помилку НЕ визиває)

.union() – Обєднує два множества // множество1.union(множество2) //

set1 = set([l, 2, 3, 4])

2 » > set2 = set([3, 4, 5, 6])

3 > » set3 = set1.union(set2)

**Другий спосіб**

**множество1 | множество2**

set3 = set1 | set2

**Пересечение множеств**

intersection() - это операция над множествами, при которой в итоговое множество входят только те элементы, которые находятся в обоих множествах.

set3 = set1.intersection(set2)

**Другий спосіб**

**множество1 & множество2**

set3 = set1 & set2

**Разность множеств**

Differenced() - ЭТО все элементы множества1, не входящие в множествe2.

set3 = set1.difference(set2)

**Другий спосіб**

**множество1 - множество2**

set3 = set1 - set2

**Симметричная разность множеств**

symmetric\_difference() - это множество, которое содержит элементы, не принадлежащие одновременно обоим исходным множествам. Иными словами, это элементы, которые входят в одно из множеств, но не входят в оба множества одновременно.

set3 = set1.symmetric \_difference(set2)

**Другий спосіб**

**множество1 ^ множество2**

set3 = set1 ^ set2

**Подмножества и надмножества**

set1 = set((l, 2, 3, 4])

set2 = set([2, 3])

В этом примере set1 содержит все элементы set2. Эго означает, что set2 является подмножеством set1. Это также означает, что setl является надмножеством set2. В Python для определения, является ли одно из множеств подмножеством другого, вызывается метод issubset ().

**issubset ()** - возвращает True, если множество2 является подмножеством множества 1. В противном случае он возвращает False.

**множество2 .issubset(множество1)**

**issuperset ()** – возвращает True, если множество1 является надмножеством множества2. В противном случае он возвращает False.

**множество1.issuperset(множество2)**

**Другий спосіб**

**множество2 <= множество1**

Данное выражение возвращает True, если множество2 является подмножеством множества1. В противном случае оно возвращает False.

**Множество1 >= множество2**

Данное выражение возвращает True, если множество! является надмножеством множества2. В противном случае оно возвращает False.

**Включение в множество**

Set1 = set([1, 2, 3, 4, 5])

set2 = {item\*\*2 for item in set1}

///

Set1 = set([1, 20, 2, 40, 3, 50])

set2 = {item for item in set1 if item < 10}

**Frozenset**

Data=frozenset([1,2,3,4,5])

З нею не можна використовувати функції по типу .pop() .update() і тп

**Створення функцій/def**

Def назв\_функ():

…

…

----------- важливо щоб було 2 enter після функції

-----------

Return щось – щоб повернути щось з функції

**Lambda функції**

Це функції які записуються в один рядок. НП

Mnogenya = lambda x, y: x\*y

Елементи які приймає що вертає

Print(mnogenya(6,6))

**Файли**

Назв = open(‘Шлях(file.txt)’, ‘режим’) – відкрити файл з якимось режимом

Назв.close()

Назв.write() – записати щось у файл

Назв.read(кількість символів) – зчитує з файла все

writelines () – для запису списків і тп (АЛЕ без \n і автоматичного ‘ ’ і все в один рядок)

Назв.readline() – читає рядок з файла (до \n) разом з \n (Если последняя строка в файле не будет заканчиваться на \п, то метод readline () вернет символьную последовательность без \п.) (повертає ‘’ коли намагається прочитати наступний рядок після останього)

Open(‘file.txt’, ‘r’, encoding=’utf-8’) – це означає що в нашому файлі можуть бути як англ так і укр мова

///

name = 'Джоанна Менчестер\п'

name = name.rstrip('\n') – видаляє вказаний символ з права

///

Режими

R – для зчитування

A – додавання, якщо в файлі вже щось є воно просто його додасть до того

W – написати щосб у файл

…+ - якщо такого файла нема то він автоматично створиться

File = open(**r**‘С :\Users\temp\test.txt’,’w’) – r ставиться для того щоб смиволи \ визначались правильно а не як НП \n або \\*

/// Пройтись по всьому файлі(по кожному рядку) з за допомогою for

File.open(‘file.txt’, ‘r’)

For line I file:

Print(line):

File.close()

////

**Винятки try/except**

Try: <------- буде виконуватись якщо немає ніяких помилок

…

Exсept назв\_помилк: <------- буде виконуватись якщо є якась помилка

……

(Можна) else: --- буде спрацьовувати якщо не спрацював except (тобто якщо try спрацював без визову помилок)

(Можна) finally: --- спрацьовую в не залежності чи є помилки чи нема

**Якщо** написати просто except: без вказування конкретної помилки то будуть опрацьовуватись всі помилки які виникають

Except назв\_помил as err: - ми присваюєм причину помилки зміній err (HП except ValueError as err: print(err) ‘сорок’)

except Exception as err: - при любих помилках присваює її err

Тип помилок:

Exception – включає в себе всі види помилок

**With … as**

With open(‘file.txt’,’r’) as file: --- автоматично буде закривати файл

Print(file.read())

**Модулі**

Вбудовані

Import назв\_модулля

Import назв as псевдонім --- тепер не по назві а по псевдонімі будем звертатись

Import назв, назв – можна підключати модулі через кому

From math import sqrt --- з math імпортуєм лише функцію sqrt

From math import sqrt as s --- переназвали функцію sqrt як s

Щоб створити власний модуль, в чистому документі.py пишем що має робити наш модуль і потім його імпортує в нашу програму

Приклади модулів:

Import time

Import datetime

Import sys

Import os

Import platform

Import math

Стороні модулі:

Щоб їх скачати вик сайт pypi.org

**ООП**

Клас і як він називається

Class Dog:

Name = None

Age = None

Ishappy = None

Def set\_data(self, nashi\_name, nash\_age):

Метод, пишеться в класі

Self обовязково щоб ми могли звертатись до НП name або age

Self.name = nashe\_name

Треба звертатись через self

Self.age = nash\_age

///

/// <------- має бути 2 enter

Об’єкт і якому класу він належить

Dog1 = Dog()

Dog1.name = “archi”

**\_\_str\_\_**

Def \_\_str\_\_(self):

Return f’….’

Main:

Print(name\_class)

Метод \_\_str\_\_() вызывается не напрямую, а автоматически во время передачи объекта в качестве аргумента в функцию print.

Метод \_\_str\_\_() также вызывается автоматически, когда объект передается в качестве аргумента во встроенную функцию str. Вот пример:

account = bankaccount2.BankAccount(1500.0)

message = str(account)

print(message)

**Конструктори**

Class Dog:

Name = None

Age = None

Ishappy = None

Встановлення за замовчуванням, тобто навіть якшо не передається ніяке значення воно всеодно буде встановлене

Def set\_data(self, nashi\_name, nash\_age=4):

Self.name = nashe\_name

Self.age = nash\_age

Конструктор, зразу при створені обєка може робити з ним щось, НП надавати якісь значення або щось виводити на екран

Def \_\_init\_\_(self, name, age):

Self.set\_data(name,age)

Dog1 = Dog(“archi”,4)

Dog1.set\_data(‘archi’)

Щоб зразу встановлювати значення для чогось іншого а не по порядку

Dog1.set\_data(age=3)

**Спадкування**

Class Build:

Year = None

City = None

Def \_\_init\_\_ (self,year,city):

Self.year = year

Self.city = city

Self.get\_info()

Def get\_info(self):

Print(…)

Class School(Build):

Pupils = None

Метод/функція щоб звернутись до головного класа

Вказуєм поточний клас і self

Def \_\_init\_\_(self, year, city, pupils):

Super(School, self).\_\_init\_\_(year, city)

Через крапку вказуєм метод або щось інше до чого нам потрібно звернутись

Self.pupils = pupils

Def get\_info(self):

Поліморфізм, це коли ми берем і переписуєм метод на інші функції

Print(self.pupils)

Super().get\_info()

Щоб викликати старий метод

School=School(1990,’kiyv’)

Home=Build(2000,’lviv’)

**Інкапсуляція**

Class Dog:

\_\_ - щоб зробити поле закритим

\_\_age = None

\_\_name = none

**Декоратори функції**

Def validator(func):

Декораток, там де beforе і after пишем наш код

Def wrapper(url):

Print(‘before’)

Func(url)

Print(‘after’)

Return wrapper

@validator

Щоб підключити наш декоратор до функції

Def open\_url(url):

Webbrowser.open(url)

**Import random**

Random.randint(1,10) – поверне число від 1 до 10 (1 і 10 включно)

Random.randrange(0,20,2) – повертає випадкове число від 0 до 19 (бо вказано 20), з проміжком 2 (тобто через одну)

Random.random() – повертає число від 0.0 до 1.0 (але крім 1.0)

Random.uniform(1.0,10.0) – повертає число від 1.0 до 10.0

Random.seed(10) – буде генерувати опрідільону послідовність випадкових чисел (замість 10 може бути інше число)

random.choice(mas) – випадково вибирає якийсь елемент з списку

random.choices (numbers, k=3) (k=кількість разів скільки потрібно взяти рандомний елемент), (весь час вибирає з списку заново,(НП [1,2,3,6,7,8]---> [8, 7, 7]))

random.sample(numbers, k=3) – без повторень таких як у choices()

**Import os**

Os.remove(‘назв’) – видаляє з діска файл з вказаною назвою

Os.rename(‘назв’,’нова\_назв’) – переменовую файл з вказаною назво на іншу вказану назву

**Import pickle**

Располагает различными функциями для сериализации, или консервации, объектов.

Import pickle

**Концервация**

1.Открывается файл для двоичной записи.

2. Вызывается метод dump() модуля p ickle, чтобы законсервировать объект и записать его в указанный файл.

3. После консервации всех объектов, которые требуется сохранить в файл, этот файл закрывается.

Обовязково щоб концервувати в режимі файла має бути ‘wb’ НП file = open(‘file.**dat**’,’**wb**’) (wb значить що він для двоичной записи) После открытия файла для двоичной записи вызывается функция dump модуля pickle. Вот общий формат функции dump:

**pickle.dump(объект, файл)**

**НП**

import pickle

phonebook = [1,2,3,4,5,6,7]

output\_file = open('phonebook.dat', 'wb')

pickle.dump (phonebook, output\_file)

output\_file.closed()

**Расконцервация**

1.Открывается файл для двоичного чтения.

2. Вызывается функция load модуля pickle, чтобы извлечь объект из файла и его расконсервировать.

3. После расконсервации из файла всех требующихся объектов этот файл закрывается.

Обовязково щоб зчитати треба режим ‘rb’

inputfile = open('mydata.dat', 'rb')

После открытия файла для двоичного чтения вызывается функция load модуля pickle. Вот общий формат инструкции, которая вызывает функцию load:

объект = pickle.load(файл)

**НП**

import pickle

file=open(‘file.dat’,’rb’)

pb=pickle.load(file)

file.close()

///

# Прочитать файл до конца.

while not end\_of\_file:

try:

# Расконсервировать следующий объект.

person = pickle.load(input\_file)

#Показать объект.

display\_data(person)

except EOFError:

# Установить флаг, чтобы обозначить, что

# был достигнут конец файла.

end\_of\_file = True

**Import math**

Функция модуля math

acos(х) - Возвращает арккосинус числа х, заданного в радианах

asin(х) - Возвращает арксинус числа х, заданного в радианах

atan(х) - Возвращает арктангенс числа х, заданного в радианах

282 Гпава 5. Функции

ceil(х) Возвращает самое малое целое, которое больше или равно х (int)

cos(х) Возвращает косинус числа х в радианах

degrees(х) Допустим, что х — это угол в радианах, тогда данная функция возвращает угол,

преобразованный в градусы

exp(х) Возвращает е

floor(х) Возвращает самое большое целое число, которое меньше или равно х (int)

hypot(х, у) Возвращает длину гипотенузы, которая проходит из (0, 0) в (х, у)

log(х) Возвращает натуральный логарифм числа х

loglO(х) Возвращает логарифм по основанию 10 числа х

radians(х) Допустим, что х — это угол в градусах, тогда данная функция возвращает угол,

преобразованный в радианы

sin(х) Возвращает синус х в радианах

sqrt(х) Возвращает квадратный корень из х

tan(х) Возвращает тангенс х в радианах

///

Math.pi – число П

Math.e – число Е

**matplotlib**

pip install matplotlib

import matplotlib

import matplotlib.pyplot as plt

pyplot – потрібен для створення всіх графіків

**.plot**(<x>,<y>) – для сворення лінійних графіків

.plot(<x>,<y>, marker='o') – добавляє точки на наших координатах

marker5='o'- Показывает круглые точки

marker5='s' - Показывает квадраты

marker5='\* ' - Показывает звездочки

marker5='D' - Показывает ромбы

тагкег5='^ ' - Показывает восходящие треугольники

marker5='v' - Показывает нисходящие треугольники

marker5='>' - Показывает левые треугольники

marker5='<' - Показывает правые треугольники

plt.plot(x,y, 'o') – створить точки але не зєднує їх лініями

**plt.bar**(<x>,<y>, width) – створює графік з столбіків де x це (так скажем) номерація стовбців, а у це їх висота, третій аргументом можна передати їх ширену

.bar(x,y,color=(‘some’)) – color передає кортеж в який можна вказати ключ кольрів для стовпців

|  |  |  |  |
| --- | --- | --- | --- |
| ‘b’ | Синій | ‘m’ | Сиреневий |
| ‘g’ | Зелений | ‘y’ | Жовтий |
| ‘r’ | Червоний | ‘k’ | Чорний |
| ‘c’ | голубий | ‘w’ | білий |

**Plt.pie**(<values>) – створює кругову діаграму де values це список з значеннями секторів

plt.pie(<mas>,labels=s) - labels= приймає масив з назвами те як ми будем підписувати наші сектори

plt.pie(<mas>,colors=(’…’)) – можна вказати свої кольри секторів (автоматично вони встановлюються так: синій, зелений, красний, голубий, сиреневий, жовтий, чорний, білий)

Plt.show() – показати графік

plt.title("career") – добавляє заголовок графіку

plt.xlabel("x\_lable") – добавляє опис х кординат

plt.ylabel("y\_label") - добавляє опис у кординат

plt.grid(True) – включає сітку на графіку

plt.xlim(xmin=-1,xmax=100) – встановлює мін і макс по х які ми будем бачити

plt.ylim(ymin=-1,ymax=50) - встановлює мін і макс по у які ми будем бачити

plt.xticks([0,1,2,3,4],['zero','one','two','three','four']) – підписуєм мітки по х

plt.yticks([0,1,2,3,4],['zero','one','two','three','four']) - підписуєм мітки по y

**Розумні і корткі рішення:**

5< =1 5> =0

Def iii(x):

Return ‘’.join(‘0’ if c<’5’ else ‘1’ for c in x)

///

Всі Т замінити на U

Return dna.replace(“T”,”U”)

Оцінки

def get\_grade(s1, s2, s3):

return {6:'D', 7:'C', 8:'B', 9:'A', 10:'A'}.get((s1 + s2 + s3) / 30, 'F')

def get\_grade(s1, s2, s3):

mean = sum([s1,s2,s3])/3

if mean>=90: return 'A'

if mean>=80: return 'B'

if mean>=70: return 'C'

if mean>=60: return 'D'

return 'F'

/////

Перевірити чи є елем в масиві

def check(seq, elem):

return elem in seq

///

Число задом на перед

def Descending\_Order(num):

return int("".join(sorted(str(num), reverse=True)))

////

Перевіти чи однаково х і о

def xo(s):

s = s.lower()

return s.count('x') == s.count('o')

///

Порахувати скільки бітів в числі

ef countBits(n):

return bin(n).count("1")