**ЛЕКЦІЯ 3**

**Приклад 3.1.Список усіх ключових слів**

>>> import keyword

>>> keyword.kwlist

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

**Приклад 3.2. Одержання списку вбудованих ідентифікаторів**

>>> import builtins

>>> dir(builtins)

['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '\_', '\_\_build\_class\_\_', '\_\_debug\_\_', '\_\_doc\_\_', '\_\_import\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'execfile', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 'runfile', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']

**Приклад зміни елемента списку**:

>>> arr = [1,2,3]

>>> arr[0] = 0 # Змінюємо перший елемент списку

>>> arr

[0, 2, З]

**ПРЕЗЕНТАЦІЯ 3**

**Приклад 1**.

size=int(input("write the number"))

if size >= 0:

print('number positive')

else:

print('number negative')

У режимі інтерпретатора ми можемо побудувати логічний вираз і одержимо в результаті False або True

**Приклад 2**

>>> size = 1

>>> size < 0

>>> size = 0

>>> size < 0

>>>size = -1

>>> size < 0

Через деякі старі проблеми, що залишилися від Python 2, логічні значення можна розглядати як числа. True 1; False 0.

**Приклад 3**

>>> True + True

>>> True – False

>>> True \* False

**Приклад 4.**

>>>type(2147483647),

type(999999999999999999999999)

**Приклад 5**.

Введення літералів у різних системах числення

# Десяткове число

>>> 123445567788778

# Двійкове число (починається з префікса 0b)

>>> 0b001101011010101010101010101

# Вісімкове число (починається з префікса 0o)

>>> 0o1237645

# Шістнадцяткове число (починається з префікса 0x)

>>> 0xdec1F

**Приклад 6.**

>>> float(5.4)

>>> float(8.9e-4)

>>> float(8.9e-12)

**Приклад 7**

>>> 2+5J

>>> 23j

>>> 11+3j, 23+12j

**Приклад 8**

Можна створити змінну типу NoneType:

my\_variable = None

*Перевірити:*

if my\_variable is None:

print('my\_variable is None')

else:

print('my\_variable is not None')

**Приклад 9**.

>>> S = "Spam"

>>> len(S) #довжина

>>> S[0] #перший елемент в S

>>> S[1] #другий елемент зліва

**Приклад10.**

>>> list1=[1,2,3,4]

>>> print(list1)

>>> list2 = [x\*\*2 for x in range(10)]

>>> print(list2)

>>> list3 = list("abcde")

>>> print(list3)

**Приклад 11.**

>>> for s in "one","two":

print(s)

#цикл за значеннями кортежу

>>>p = (1.2, 3.4, 0.9) #точка в тривимірному просторі

>>> print(p)

>>> p1 = 1, 3, 9 #без дужок

>>> print(p1)

>>> p2 = 1, 2, 3, 4, #кома ігнорується

>>>print(p2)

**Приклад 12**

>>>#one parameter

>>> for i in range(5):

print(i)

>>> # Two parameters

>>> for i in range(3, 6):

print(i)

>>> # Three parameters

>>> for i in range(4, 10, 2):

print(i)

>>> # Going backwards

>>> for i in range(0, ‐10, ‐2):

print(i)

**Приклад 13**

dan={'Name':'Ivan','Age':17,'Course':1}

print(dan)

print('Name: ',dan['Name'])

------

{'Course': 1, 'Age': 17, 'Name': 'Ivan'}

Name: Ivan

**Приклад 14**

Group={"Galenko","Petrenko","Sydorko"}

Total=len(Group)

print(Group)

print("Total: ",Total)

if " Petrenko " in Group:

print("Petrenko is a student")

Group.add("Trump")

print(Group)

------

**Приклад15**.

Зміна елемента списку (тип list)

>>> arr = [1,2,3]

>>> arr[0]=0 #змінюємо перший елемент списку

>>> arr [0, 2, 3]

**Приклад 16**

>>> strl = "авто"

>>> str2 = "транспорт"

>>> str3 = strl + str2 # Конкатенація

>>> print(str3)

**Приклад 17.**

Вивести елементи списку можна так:

>>> arr = [1, 2, 3]

>>> i = iter(arr)

>>> next(i)

1

**Приклад 18**.

d = {"х": 1, "y": 2}

i = iter(d)

c = next(i)

print(c,':',d[c])

c = next(i)

print(c,':',d[c])

-----

х : 1

y : 2

**Приклад 19.** Вивід елемента списку:

>>> for i in [1, 2]:

print(i)

**Приклад 20.** Перебрати слово по буквах:

>>> for i in "Рядок": print(i)

**Приклад 21.** Перебір елементів словника:

>>> d = {"х": 1, "y": 2}

>>> for key in d:

print(d[key])

1

2

**Приклад 22.** Чому не можна застосовувати групове присвоєння для змінюваних об'єктів

>>> х = y = [1, 2] #Наче створили два об'єкти

>>> х, y

У цьому прикладі ми створили список із двох елементів і присвоїли значення змінним х и y. Тепер спробуємо змінити значення в змінній y:

>>> y[1] = 100 # Змінюємо другий елемент

>>> х, y

Як видно з прикладу, зміна значення в змінній y привела також до зміни значення в змінній х. Таким чином, обидві змінні посилаються на той самий об'єкт, а не на два різні об'єкти. Щоб одержати два об'єкти, необхідно робити роздільне присвоювання:

>>> х = [1, 2]

>>> y = [1, 2]

>>> y[l] = 100 # Змінюємо другий елемент

>>> х, y

**Приклад 23** Перевірка на одне і те ж посилання

>>> x = y = [1,2] #один об'єкт

>>> x is y +

>>> x = [1,2] #різні об'єкти

>>> y = [1,2] #різні об'єкти

>>> x is y -

**Приклад 24**.

>>> х = 2; y = 2; z = 2 +

>>> х is y, y is z

**Приклад 25**. Позиційне присвоювання:

>>> х, y, z = 1, 2, 3

>>> х, y, z

**Приклад 26.**

>>> х, y = 1, 2;

>>>х, y

>>> х, y = y, х;

>>> х, y

**Приклад 27**

>>> х, y, z = ”123” # Рядок

>>> х, y, z

>>> х, y, z = [1, 2, 3] # Список

>>> х, y, z

>>> х, y, z = (1, 2, 3) # Кортеж

>>> х, y, z

>>> [х, y, z] = (1, 2, 3) #Список ліворуч, кортеж праворуч

>>> х, y, z

**Приклад 28**

>>>х,y,\*z=(1,2,3,4)

>>>х,y,z

>>>х,\*y,z = ( 1,2,3,4)

>>>х,y,z

>>>\*х,y,z =(1,2,3,4)

>>>х,y,z

>>>х,y,\*z =(1,2,3)

>>>х,y,z

>>>х,y\*z=1,2

>>>х,y,z

**Приклад 29.**

>>> a = 'Рядок' #тип str

>>> a = 7 #тепер змінна має тип int

**Приклад 30.**

>>>bool(0),bool(1),bool(""),bооl("Рядок"),bool([l,2])

>>>bool([])

**Приклад 31**:

>>> int(7.5),int("71")

>>>int("71",10),int("71",8),int("0o71",8),int("A",16)

**Приклад 32**:

>>> float(7),float("7.1")

>>> float("Infinity"), float("-inf")

>>> float("Infinity") + float("-inf")

**Приклад 32**:

>>> str(l25), str([l, 2, 3]))

>>> str((1,2,3)), str({"x": 5, "в": 10})

>>> str( bytes("а", "UTF-8") )

>>> str( bytes("F", "UTF-8") )

>>> str( bytearray("З", "UTF-8") )

>>> str( bytearray("L", "UTF-8") )

**Приклад 33**:

>>> list("12345") #Рядок str⇒ list

>>> list("Рядок") #Рядок str⇒ list

>>> list({1,2,3,4,5}) #Множина set⇒ list

>>> list({"x": 5, "y": 10}) #Словник dict⇒ list

**Приклад 34**

>>> tuple("123456") #Рядок str⇒ tuple

>>>tuple([1,2,3,4,5]) #Список list⇒ tuple

>>> tuple(({"x": 5, "y": 10})) #Словник dict⇒ tuple

**Приклад 35**.

х = input("x = ") # Уводимо 5

y = input("y = ") # Уводимо 12

print (х + y)

Результатом виконання цього скрипта є не число, а рядок 512. Таким чином, слід запам’ятати, що функція input () повертає результат у вигляді рядка. Щоб додати два числа, необхідно перетворити рядок на число

**Приклад 36. Перетворення рядка в число**

х = int(input("x = ")) # Вводимо 5

y = int(input("y = ")) # Вводимо 12

print (х + y)

17

**Приклад 37. Видалення однієї змінної:**

>>> x=20;

>>>x

20

>>> del x

>>>x

Traceback (most recent call last):

File «<input”> line 1, in <module>

Nameerror: name 'x' is not defined

**Приклад видалення декількох змінних**:

>>> х, y = 10, 20

>>> del х, y

**ЛЕКЦІЯ 4**

**Приклад 1.**

>>> 10 // 5 # Ділення цілих чисел без остачі

>>> 10 // 3 # Ділення цілих чисел з остачею

>>> -10 // 3 # Ділення цілих чисел з остачею

>>> 10.0 // 5.0 # Ділення дійсних чисел

>>> 10.0 // 3.0 # Ділення дійсних чисел

>>> -10 // 5.0 # Ділення цілого числа на дійсне

>>> 10 // 3.0 # Ділення цілого числа на дійсне

**Приклад 2.**

>>> 10 % 5 # Ділення цілих чисел без остачі

0

>>> 10 % 3 # Ділення цілих чисел з остачею

>>> 10.0 % 5.0 # Операція над дійсними числами

>>> 10.0 % 3.0 # Операція над дійсними числами

>>> 10%5.0 # Операція над цілими й дійсними числами

>>> 10%3.0 # Операція над цілими й дійсними числами

**Приклад 3.**

>>> 0.3-0.1-0.1-0.1

-2.7755575615628914e-17

Очікуваним був би результат 0.0, але, як видно з прикладу, ми одержали зовсім інший результат. Якщо необхідно виконувати операції з фіксованою точністю, то слід використовувати модуль decimal:

**Приклад 4.**

>>> from decimal import Decimal

>>> Decimal("0.3")-Decimal("0.1")-Decimal("0.1")-Decimal("0.l")

Decimal('0.0')

**Приклад 5.**

>>> х = 0b10

>>> х = ~х

>>> bin(x)

>>> х = -0b11

>>> х = ~х

>>> bin(x)

**Приклад 6.**

>>> х = 0b101

>>> y = 0b110

>>> z = х & y

>>> bin(z)

>>> "{0:b} & {1:b} = {2:b}". format(х, y, z)

'101 & 110 = 100'

| — двійкове OR

**Приклад 7.**

>>> х = 0b101

>>> y = 0b110

>>> z = х | y

>>> bin(z)

>>> "{0:b}|{1:b}={2:b}".format(x, y, z)

'101 | 110 = 111'

^ — операція XOR:

**Приклад 8.**

>>> х = 0b001

> > > y = 0b111

>>> z = х ^ y

>>> bin(z)

'0b110'

>>> "{0:b}^{1:b}={2:b}".format(x, y, z)

’1 ^ 111 = 110'

**Приклад 9.**

>>> х = 0b1111 # Введення x

>>> y = х << 1 # Зсув на 1 розряд

>>> bin(y) '

0b11110'

>>> z = y << 1 # Зсув на 1 розряд

>>> bin(z)

'0b111100'

>>> k = z << 2;

>>> bin(k) # Зсув на 2 розряди

'0b11110000'

**Приклад 10.**

**Додатне число**

>>> x = 0b11

>>>y=x>>1; bin(y)

'0b1'

**Від’ємне число**

>>> x=-0b11

>>> y=x>>1; bin(y)

'-0b10'

**Приклад 11.**

>>> 2 not in [1,2,3], 4 not in [1,2,3] # список

>>> 2 not in [1,2,3]; 4 not in [1,2,3] # список

>>> 2 not in (1,2,3), 6 not in (1,2,3) #кортеж

**Приклад 12.**

>>> х = 10; х //= 3 # Еквівалентно х = х // 3

>>> х

>>> х = -10; х //= 3 # Еквівалентно х = х // 3

>>> х

>>> y = 10.0; y //= 3.0 # Еквівалентно y=y//3.0

>>> y

**Приклад 13. MOD**

>>> х = 10; х %= 2 # Еквівалентно х = x % 2

>>> х

>>> y = 10; y %=3 # Еквівалентно y = y % 3

>>> y

**\*\* = - піднесення до степеня і присвоювання:**

>>> х = 10; х \*\*= 2 #Еквівалентно x = x \*\* 2

>>> х

**Приклад 14.** tru

>>> bool(l), bоо1(20), bool(-20)

>>> bool(1.0), bool(0.1), bool(-20.0)

**не порожній об'єкт**:

>>> bool("0"), bool([0,None])

>>> bool((None,)), bool({"x":5})

**Приклад 15** число, що дорівнює нулю: fal

>>> bool(0), bool(0.0)

(False, False)

порожній об'єкт:

>>> bool(""), bool([]), bool(())

(False, False, False)

значення None:

>>> bool(None)

False

ПРИКЛАД 16

**is not** - перевіряє, чи посилаються дві змінні на різні об'єкти

Якщо це так, повертається значення True:

>>> х = y = [ 1, 2)

>>> х is not y

>>> х = [ 1, 2] ; y = [1, 2]

>>> х is not y

Значення логічного виразу можна інвертувати за допомогою оператора **not**:

>>> х = 1; y = 1

>>> х == y

>>> not (х == y), not х == y

**Приклад 17**

>>> х = 10

>>> 1 < х < 20, 11 < х < 20

Кілька логічних виразів можна об'єднати в один великий за допомогою наступних операторів: **аnd** - логічне **AND**. Якщо **х** у виразі **х** **and** **y** інтерпретується як **False**, то повертається **х**, а якщо ні, то - **y**:

**Приклад 18**

>>> 1 < 5 and 2 < 5 True and True == True

>>> 1 > 5 and 2 < 5 #false and True == False

>>> 10 and 20, 0 and 20, 10 and 0

**Приклад 19**

>>> 1 < 5 or 2 < 5 # True or True == True

>>> 1 < 5 or 2 > 5 # True or False == True

>>> 1 > 5 or 2 < 5 # False or True == True

>>> 1 > 5 or 2 > 5 # False or False == False

>>> 10 or 20, 0 or 20, 10 or 0

**ПРЕЗЕНТАЦІЯ 4**

**Приклад 11.**

>>> "Рядок" not in "Рядок для пошуку" # Рядки

>>> "Рядок2" not in "Рядок для пошуку" # Рядки

**>>> 2 not in [1,2,3], 4 not in [1,2,3] # список**

**>>> 2 not in [1,2,3]; 4 not in [1,2,3] # список**

>>> 2 not in (1,2,3), 6 not in (1,2,3) #кортеж

**Приклад 18** **логічне AND.**

>>> 1 < 5 and 2 < 5 True and True == True

>>> 1 > 5 and 2 < 5 #false and True == False

>>> 10 and 20, 0 and 20, 10 and 0

**Приклад 19 логічне OR**

>>> 1 < 5 or 2 < 5 # True or True == True

>>> 1 < 5 or 2 > 5 # True or False == True

>>> 1 > 5 or 2 < 5 # False or True == True

>>> 1 > 5 or 2 > 5 # False or False == False

>>> 10 or 20, 0 or 20, 10 or 0

(10, 20, 10)