Зовнішні модулі. Вбудовані функції. Власна бібліотека.

Зовнішні модулі

Окрім вбудованих бібліотек в Python можна завантажити *зовнішні модулі.* Вони існують для більш швидкого та легкого досягнення окремих цілей. Наприклад, модуль cv2 існує для роботи з веб-камерою, flask – для веб-розробки та т.і.

Для завантаження зовнішніх модулів можна скористатись файловим менеджером. Найпоширенішім для таких цілей є *pip*. Після його завантаження на ПК, для встановлення будь-якої бібліотеки достатньо лише в консолі операційної системи написати команду:

pip install *назва модуля*

та дочекатись завершення завантаження.

Після загрузки бібліотеки на комп'ютер її можна імпортувати у python файл як звичайний модуль.

Корисні вбудовані функції

Розглянемо деякі функції для роботи з символами та деякі математичні функції, що знаходяться не в модулі math.

Функція *ord()* (Order) приймає в якості аргумента один символ в лапках та повертає його номер у таблиці Unicode.

```
>>> ord('r')
114
>>> ord('9')
57
>>> ord('%')
37
>>> ord('y')
1091
```

Функція *chr()* (Character) приймає в якості аргумента ціле число та передає символ з таблиці Unicode за цим номером.

```
>>> chr(10)
'\n'
>>> chr(114)
'r'
>>> chr(37)
'%'
```

Функція len(), окрім масивів, може також приймати строки. Результатом буде кількість символів у данному рядку.

```
>>> len('qw_er ty')
8
>>> a = 'string'
>>> len(a)
6
```

Щодо математичних функцій, не завди доцільно викликати модуль math, так як існують вбудовані аналоги функцій.

Функція *abs()* приймає число на повертає його абсолютне значення, тобто його значення по модулю.

```
>>> abs(-17.51)
17.51
>>> abs(17.52)
17.52
```

Функція *round()* округляє отримане число до вказаного знаку. Першим аргументом функція приймає число, другим – знак після дробу, до якого буде округлення. Якщо аргумент буде всього один, то число буде округлятися до цілої частини. Також другий аргумент може бути від'ємним – тоді число буде округлятися до десятків/сотень/тисяч/т.д.

```
>>> round(12.358936, 2)
12.36
>>> round(-17.94)
-18
>>> round(15743, -2)
15700
```

Функція *divmod()* приймає два числа та повертає кортеж з двох елементів – результта ділення націло першого аргумента на другий та залишок від їх ділення.

```
>>> divmod(27, 5)
(5, 2)
```

Аналогом цієї функції є два оператора: // (ділення націло) та % (залишок від ділення націло).

```
>>> 27//5
5
>>> 27%5
2
```

Функції max() та min() можуть працювати з нескінченною кількістю аргументів. Перша повертає серед них найбільший, друга – найменший. При цьому обидві ці функції можуть приймати строки як аргумент. У такому випадку сортування відбуватиметься не за значенням, а за алфавітом. Також в функції можливо передавати переліки елементів. У цьому випадку результатом є відповідний елемент масиву.

```
>>> max(4,2,12)
12
>>> min(4,2,12)
2

>>> max('abc', 'bcdef', 'f')
'f'
>>> min('100', '1', '01')
'01'
```

```
>>> s = [1,2,3,4,5]
>>> max(s)
5
>>> min(s)
1
```

Функція *sum()* може отримувати тільки перечень елементів і тільки з числами. Повертає функція суму всіх елементів переданного масиву.

```
>>> s = [1,2,3,4,5]
>>> p = (0, 12, 23, 20)
>>> sum(s)
15
>>> sum(p)
55
```

Створення власної бібліотеки

Для створення власного модулю робимо файл з розширенням *.py*, який буде складатись тільки з функцій. Назва цього файлу – це назва , за якою її можна імпортувати. а назви функцій, записаних у цьому файлі – це назви функцій модулю. Після того, як наша бібліотека буде цілком готова, зберігаємо її в ту саму папку, де знаходиться інтерпретатор мови Python (там же знаходяться всі інші бібліотеки) або у папку проекта (в цьому випадку цим модулем можна користуватися лише в рамках директорії проекту).

Власні бібліотеки можна імпортувати до файлу так само, як і вбудовані додаткові бібліотеки.

Контрольні питання:

1. Як завантажити зовнішній модуль?
2. Що робить функція round()? Що є її аргументами?
3. Що буде, якщо помістити рядок аргументом функції len()?
4. Навіщо створювати власну бібліотеку?