
Яндекс

Академия Яндекса

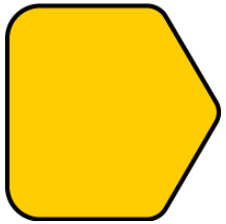
Функции.
Возвращение значений из функций

Связь между математическими функциями и функциями в Python

Связь между математическими функциями и функциями в Python

Каждая функция может не только выполнять действия, но и выдавать какой-то результат, который потом можно использовать в программе — например, записать в переменную.

Вы еще не делали таких функций, но уже не раз пользовались ими. Попробуйте вспомнить несколько примеров.

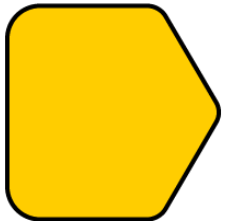


Функция в математике — такое преобразование, которое из одного значения или набора значений делает другое значение.

$$f(a,b) = (a+b)^2 \qquad f(x) = \sqrt{x} \qquad f(R) = \frac{4}{3}\pi R^3$$

Связь между математическими функциями и функциями в `Python`

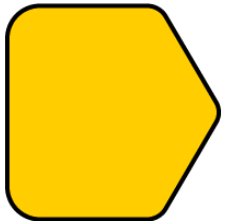
Заметьте: математическая функция не обязана работать с числами. Например, в математике можно встретить такую функцию — число перестановок букв в слове. Это функция, которая принимает аргументом строку, а возвращает число. Или функцию пересечения множеств, которая берет в качестве аргументов два множества и возвращает тоже множество.



Функция в языке `Python` — некий алгоритм, который выполняется каждый раз одинаково. Для большинства функций возвращаемое значение, как и в случае математической функции, зависит только от аргументов.

Функции в `Python`

У функций в `Python`, как вы знаете, тоже есть список аргументов: иногда одно значение, иногда несколько, а иногда он и вовсе пустой, как у функции `input`. У функций также есть возвращаемое значение — значение всегда ровно одно.



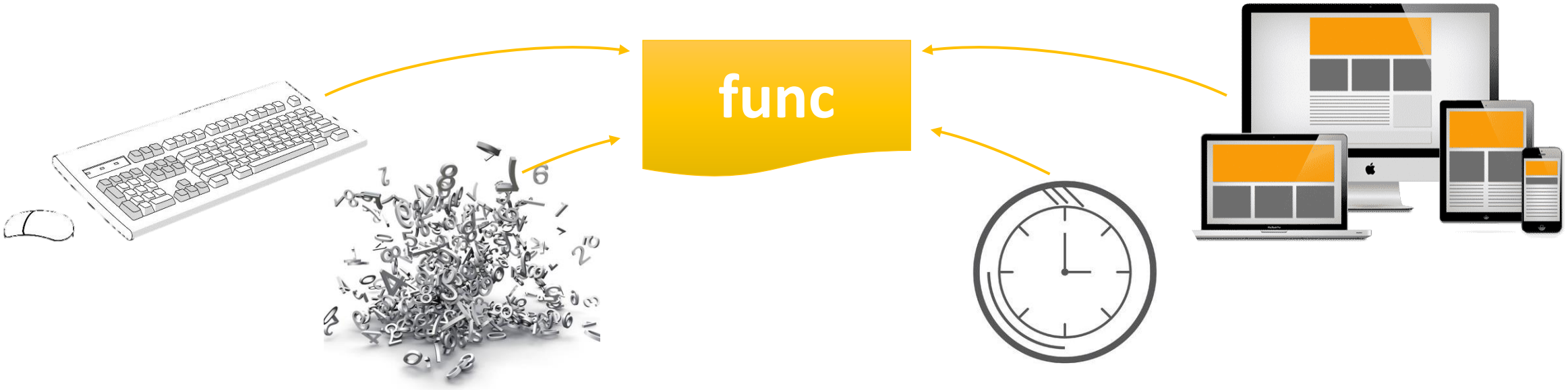
Важно!

Может показаться, что некоторые функции ничего не возвращают (как функция `print`), но на самом деле они тоже возвращают значение — `None`.

Функции в `Python`

Их отличие от математических функций в том, что функция в `Python` может зависеть не только от аргументов, но и от внешних причин. Например, от действий пользователя.

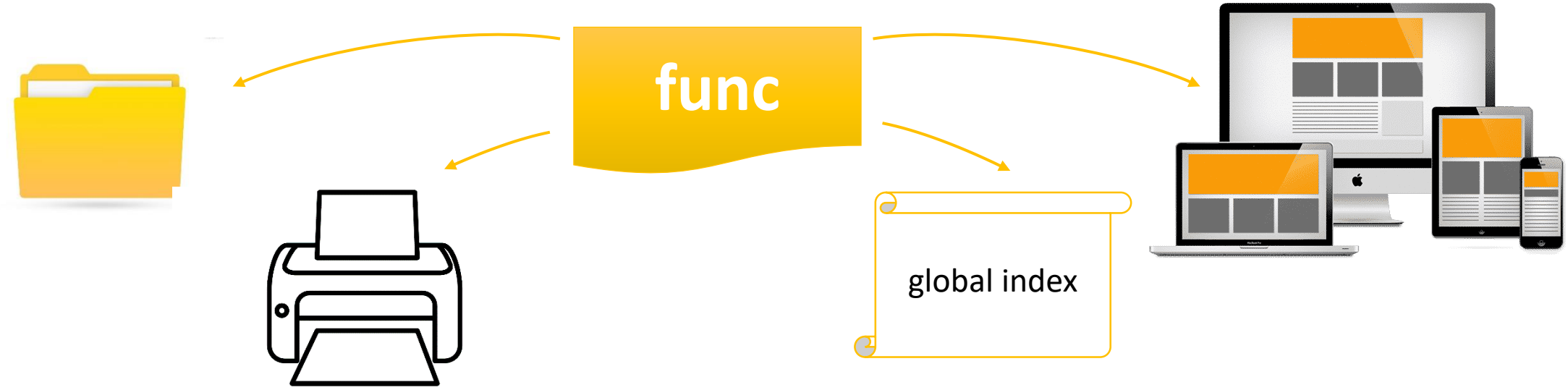
Функция `input`, помимо пустого списка аргументов, получает ввод с клавиатуры пользователя. Некоторые функции читают файлы на жестком диске или в Интернете, а значит, их результат зависит от содержимого файла или веб-страницы. Есть функции, работа которых зависит от текущего времени. Есть функции, зависящие от датчика случайных чисел.



Функции в Python

Кроме того, работа некоторых функций зависит от внешних (глобальных) переменных. Это еще одна особенность функций в Python: они могут изменять что-то снаружи функции — менять глобальные переменные, выводить текст на экран, записывать что-то в файлы.

Таким образом, функции можно разделить на функции с побочными эффектами и без них.



Функции в Python

Функции можно разделить на функции с побочными эффектами и без них.

Функции без побочных эффектов

- «чистые» функции
- ведут себя в точности как математические функции
- math.sqrt, math.cos, abs, int, str, len, min, max

Функции с побочными эффектами

На следующем уроке

- функции, предназначенные для общения с «внешним миром»: пользователем, файлами на жестком диске, другими программами или серверами в Интернете
- input и print

Возвращаемые значения

Возвращаемые значения

Для того чтобы функция вернула значение, используется оператор **`return`**. Использовать его очень просто. Давайте напишем функцию **`double_it`**, которая удваивает значение:

```
def double_it(x):
    return x * 2
```

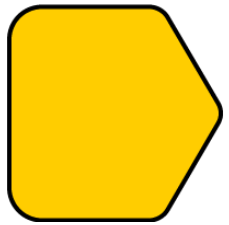
Эта функция получила число x в качестве аргумента, умножила его на 2 и вернула результат в основную программу. Значением, которое функция вернула, можно воспользоваться. Например, мы можем посчитать длину окружности с использованием этой функции:

```
radius = 3
length = double_it(3.14) * radius
```

Когда интерпретатор дойдет до **`double_it(3.14)`**, начнет исполняться код функции. Когда он дойдет до слова **`return`**, значение, которое указано после **`return`**, будет подставлено в программе вместо вызова функции. Можно сказать, что сразу после того как функция досчитается, вычисление превратится в такое:

```
length = 6.28 * radius
```

Возвращаемые значения



Важно!

Заметьте: функция `double_it` ничего не выводит на экран. Она выполняет вычисления и сообщает их не пользователю, как мы делали раньше, а другой части программы.

Если нам потребуется не просто вернуть удвоенное число, а еще и вывести его на экран, лучше не добавлять `print` внутрь функции. Ведь если вы добавите `print` в функцию, уже никак не сможете вызвать эту функцию в «тихом» режиме, чтобы она ничего не печатала. Вместо этого лучше сначала вернуть результат, а потом уже распечатать его во внешней программе. Вот так:

```
print(double_it(3.14))
```

Или, если вам нужно еще как-то использовать вычисленное значение, можно завести специальную переменную, хранящую результат вычисления.

```
double_pi = double_it(3.14)
print(double_pi)
length = double_pi * radius
```

Возвращаемые значения

А теперь давайте рассмотрим чуть более сложный пример — вычисление суммы элементов списка:

```
def my_sum(arr):
    result = 0
    for element in arr:
        result += element
    return result

print(my_sum([1, 2, 3, 4]))
```

Вспомогательная переменная

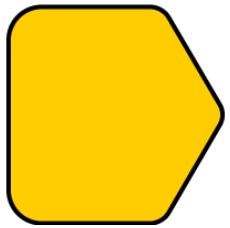
Возвращаемое значение

Здесь мы создаем вспомогательную переменную, а затем возвращаем ее значение.

Объект (значение) может существовать, даже когда нет переменной, в которой он хранится. Когда мы записываем число в **`result`**, мы фактически создаем объект числа и даем ему временное имя **`result`**. Потом, когда мы пишем **`return result`**, мы возвращаем не переменную, мы возвращаем объект «число 10».

Сборка мусора

После того как функция `print` отработала, доступ к результату вычисления пропал, ведь мы никуда не сохранили этот результат. На объект «число 10» нет ссылок, поэтому интерпретатор может его «выкинуть».



Это называется «сборка мусора»: `Python` автоматически избавляется от всех объектов, которые невозможно использовать.

Возвращение значений встроенными функциями

Возвращение значений встроенными функциями

Как узнать, значение какого типа сейчас хранится в переменной?
Для этого есть встроенная функция **`type`**:

```
a = 15
b = "hello"
print(type(a), type(b))  # <class 'int'> <class 'str'>
print(type(a) is int, type(b) is list)  # True, False
```

Возвращение значений встроенными функциями

Проверьте себя. Всегда ли функции возвращают значения одного и того же типа? Какие значения вернут эти функции?

Встроенная функция	Тип возвращаемого значения	Как проверить
len()		print(type(len("Hello")) is ?)
max()		print(type(max(["Hello", "hello"])) is ?) print(type(max([2.5, 3.8])) is ?)
int()		print(type(int("185")) is ?)
abs()		print(type(abs(-3)) is ?) print(type(abs(-1.5)) is ?)
bool()		print(type(bool([])) is ?)
print()		print(print() is ?)


Яндекс