

Python Starter

Функції

Python Starter

План заняття

1. Поняття функції
2. Види параметрів
3. Іменовані параметри під час виклику функції
4. Значення аргументів за замовчуванням (опціональні параметри)
5. Функції від невідомої кількості аргументів

Python Starter

Після уроку обов'язково



Повторіть цей урок у відео форматі на [ITVDN.com](http://itvdn.com)

Доступ можна отримати через керівництво вашого навчального центру



Перевірте, як Ви засвоїли цей матеріал на [TestProvider.com](http://testprovider.com)

Python Starter

Введення в Python

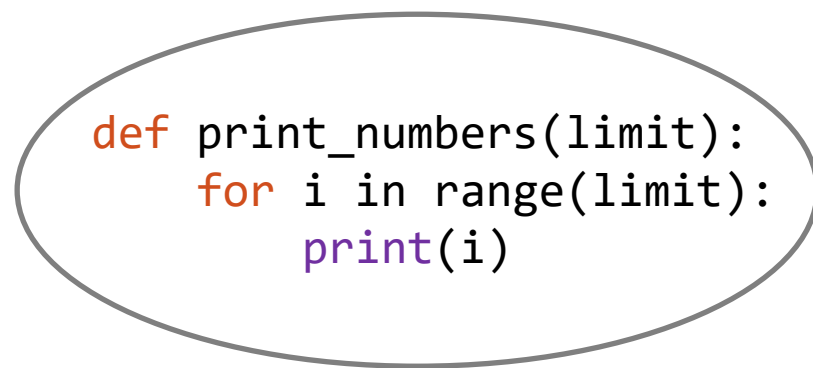
Функції

Python Starter

Поняття функції

Функція — це іменована ділянка коду, до якої можна неодноразово звертатися з інших місць програми.

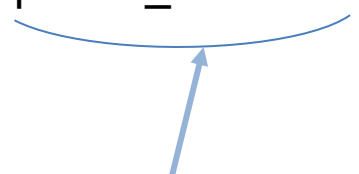
Функції можуть приймати параметри (аргументи) та повертати значення.



```
def print_numbers(limit):  
    for i in range(limit):  
        print(i)
```

оголошення
функції

```
n = int(input())  
print_numbers(n)
```



виклик
функції

Python Starter

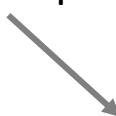
Передача параметрів

Функції можуть приймати параметри (аргументи).

Формальні параметри – параметри, що вказуються під час оголошення функції.

Фактичні параметри – параметри, які передаються у функцію під час виклику.


формальний
параметр



```
def print_numbers(limit):  
    for i in range(limit):  
        print(i)
```

```
n = int(input())  
print_numbers(n)
```

фактичний
параметр



Python Starter

Повернення результату

Деякі мови мають чіткий поділ між підпрограмами, що повертають результат (*функціями*), і тими, які ніякого результату не повертають (*процедурами*).

У Python будь-яка підпрограма є функцією. Якщо функція явно не повертає жодного значення, автоматично повертається **None**.

Для повернення значення використовується оператор **return**. Робота функції при цьому завершується. Якщо не вказати значення, яке потрібно повернути, автоматично повертається None і це можна використовувати для дострокового виходу з процедур.

```
def add_numbers(a, b):  
    return a + b
```

Python Starter

Створення функції

1. Слід обрати відповідне ім'я для функції, продумати, які параметри повинна приймати функція (від яких значень вона залежить) і що має повертати.
2. Методи визначаються за допомогою ключового слова **def**, за яким слідує *сигнатура функції* – її ім'я та перелік формальних параметрів.
3. Імена параметрів функції перераховуються в круглих дужках через кому. Якщо функція не приймає жодних параметрів, порожні круглі дужки все одно вказуються.
4. Якщо функція повинна щось повертати, це робиться за допомогою ключового слова **return**. Якщо воно не присутнє у функції або хоча б одній можливій її гілці виконання, функція автоматично поверне **None**.

Python Starter

Виклик функції

1. Функція викликається шляхом вказівки її імені та списку фактичних параметрів, розділених комами, у круглих дужках.

Важливо: якщо функція вимагає передачі її аргументів, все одно вказуються порожні круглі дужки, оскільки саме вони є ознакою виклику *функції*. Якщо цього не робити, ми отримаємо об'єкт – *саму функцію*.

2. Виклик функції – вираз. Його значення збігається зі значенням, яке повернула функція, і його можна присвоїти змінній, використовувати як аргумент іншої функції або використовувати всередині іншого виразу.
3. Якщо функція нічого не повертає або її результат нам не цікавий, можна нічому не привласнювати її результат і використовувати виклик цієї функції як головний оператор у даному рядку коду.

Python Starter

Виклик функції з іменованими параметрами

- При виклику функції фактичні параметри заміщують формальні у тому порядку, в якому вони вказані.
- Можна змінити цей порядок, вказавши під час виклику імена відповідних формальних параметрів:

`ім'я_функції(аргумент_2=значення, аргумент_1=значення)`

- При виклику функції можна використовувати іменовані та неіменовані аргументи одночасно. У такому випадку спочатку вказуються неіменовані фактичні параметри, які заміщують формальні в тому порядку, в якому в заголовку функції описано відповідні формальні параметри, а потім вказуються інші фактичні параметри разом з іменами відповідних формальних параметрів у довільному порядку.

Python Starter

Опціональні параметри

- Частину параметрів функції можна зробити необов'язковими для передачі під час виклику.
- Для цього необхідно вказати значення цих параметрів за замовчуванням, які будуть використані в тому випадку, якщо під час виклику відповідні фактичні параметри не вказані.
- Значення за замовчуванням створюються лише один раз під час створення функції та зв'язуються з іменами відповідних формальних параметрів. Тому потрібно бути обережним при використанні значень за замовчуванням, які є об'єктами, що змінюються, так як їх можна змінити для всіх наступних викликів функції. З незмінюваними об'єктами, такими як числа та рядки, цієї проблеми немає.

Python Starter

**args*

Звичайна функція:	З використанням *args:
<pre>def adder(x, y, z): print("sum:", x + y + z) adder(10, 12, 13) # sum: 35</pre>	<pre>def adder(*nums): sum = 0 for n in nums: sum += n print("Sum: ", sum) adder(3, 5) adder(4, 5, 6, 7) adder(1, 2, 3, 5, 6) # Sum: 8 # Sum: 22 # Sum: 17</pre>

Python Starter

****kwargs**

За аналогією з `*args` ми використовуємо `**kwargs` для передачі змінної кількості іменованих аргументів. Схоже з `*args`, якщо поставити `**` перед ім'ям, це ім'я прийматиме будь-яку кількість іменованих аргументів. Словник із кількох переданих аргументів буде доступний під цим ім'ям. Наприклад:

```
def intro(**data):  
    print("\nData type of argument: ", type(data))
```

```
    for key, value in data.items():  
        print("{} is {}".format(key, value))
```

```
intro(Firstname="Sita", Lastname="Sharma", Age=22, Phone=1234567890)  
intro(Firstname="John", Lastname="Wood", Email="johnwood@nomail.com", Country="Wakanda", Age=25,  
Phone=9876543210)
```

Дивіться наші уроки у відео форматі

ITVDN.com



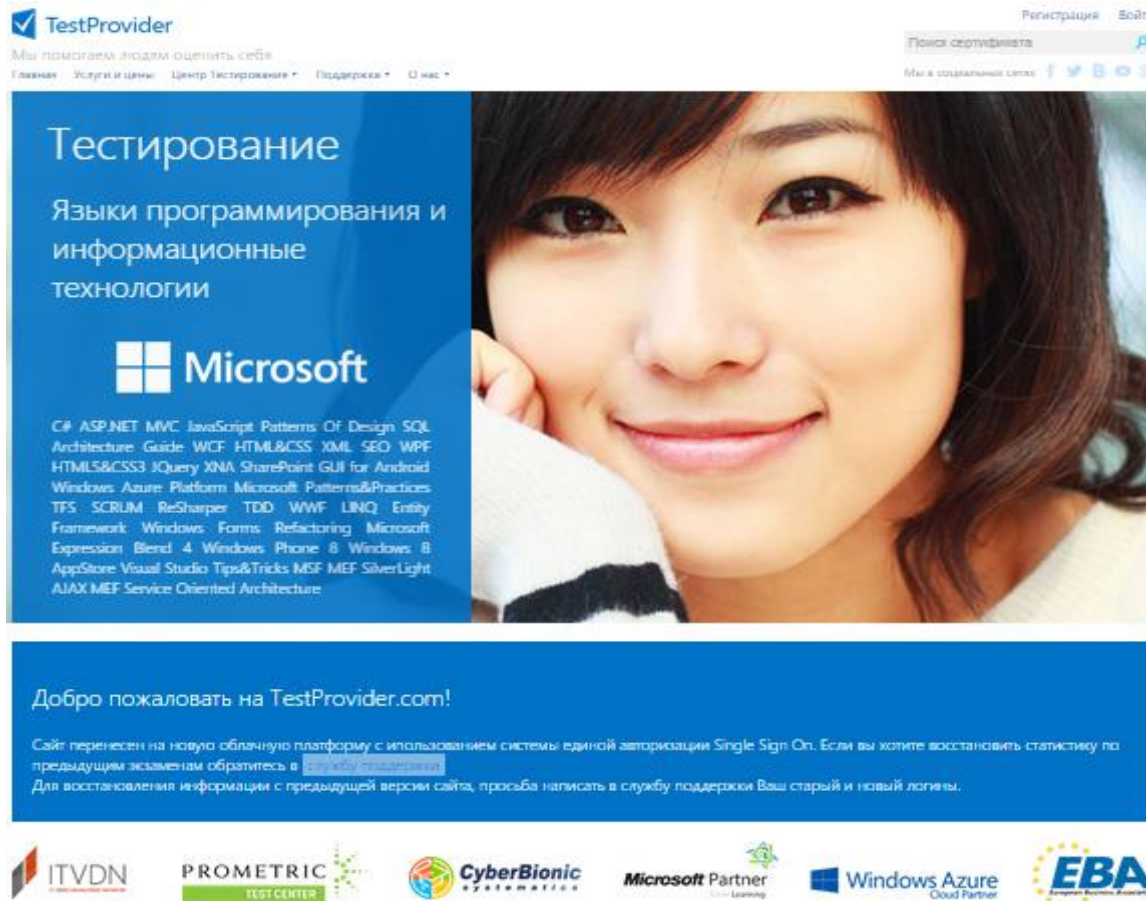
Перегляньте цей урок у відео форматі на освітньому порталі [ITVDN.com](http://itvdn.com) для закріплення пройденого матеріалу.

Усі курси записані сертифікованими тренерами, які працюють у навчальному центрі CyberBionic Systematics



Перевірка знань

TestProvider.com



TestProvider

Мы помогаем людям оценить себя

Главная Услуги и цены Центр Тестирования Поддержка О нас

Регистрация Войти

Поиск сертификата

Мы в социальных сетях

Тестирование

Языки программирования и информационные технологии

Microsoft

C# ASP.NET MVC JavaScript Patterns OF Design SQL Architecture Guide WCF HTML&CSS XML SEO WPF HTML5&CSS3 JQuery XNA SharePoint GUI for Android Windows Azure Platform Microsoft Patterns&Practices TFS SCRUM ReSharper TDD WWF LINQ Entity Framework Windows Forms Refactoring Microsoft Expression Blend 4 Windows Phone 8 Windows 8 AppStore Visual Studio Tips&Tricks MSF MEF SilverLight AJAX MEF Service Oriented Architecture

Добро пожаловать на TestProvider.com!

Сайт перенесен на новую облачную платформу с использованием системы единой авторизации Single Sign On. Если вы хотите восстановить статистику по предыдущим экзаменам обратитесь в [службу поддержки](#). Для восстановления информации с предыдущей версии сайта, просба написать в службу поддержки Ваш старый и новый логины.

ITVDN PROMETRIC TEST CENTER CyberBionic Microsoft Partner Windows Azure Cloud Partner EBA

TestProvider – це online сервіс перевірки знань з інформаційних технологій. За його допомогою Ви можете оцінити Ваш рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT спеціаліста.

Після кожного уроку проходите тестування для перевірки знань на TestProvider.com

Успішне проходження фінального тестування дозволить Вам отримати відповідний Сертифікат.



Python Starter

Q&A

Python Starter

Дякую за увагу!

Інформаційний відеосервіс для розробників програмного забезпечення

