No in-built switch statement here

How to implement a switch-case statement in Python

October 24, 2017 Sreeram Sceenivasan

#python (https://jaxenter.com/tag/python) #tutorials (https://jaxenter.com/tag/tutorial)

reddit (http://www.reddit.com/submit?url=https://jaxenter.com/implement-switch-case-statement-python-138315.html)
Twitter (https://twitter.com/home?status=How to implement a switch-case statement in Python https://jaxenter.com/implement-switch-case-statement-python-138315.html)
LinkedIn (https://www.linkedin.com/shareArticle?mini=true&url=ht switch-case-statement-python-138315.html)
Facebook (https://www.facebook.com/sharer/sharer.php?
Google+ (https://plus.google.com/share?
nt-

Switch-case statements are a powerful tool for control in programming. In this article, Sreeram Sceenivasan goes over you can use a switch-case statement in Python.

© Shutterstock / LoopAll

Switch-case statement is a powerful programming feature that allows you control the flow of your program based on the value of a variable or an expression. You can use it to execute different blocks of code, depending on the variable value during runtime. Here's an example of a switch statement in Java.

```
public static void switch_demo(String[] args) {

        int month = 8;
        String monthString;
        switch (month) {
            case 1:  monthString = "January";
                     break;
            case 2:  monthString = "February";
                     break;
            case 3:  monthString = "March";
                     break;
            case 4:  monthString = "April";
                     break;
            case 5:  monthString = "May";
                     break;
            case 6:  monthString = "June";
                     break;
            case 7:  monthString = "July";
                     break;
            case 8:  monthString = "August";
                     break;
            case 9:  monthString = "September";
                     break;
            case 10: monthString = "October";
                     break;
            case 11: monthString = "November";
                     break;
            case 12: monthString = "December";
                     break;
            default: monthString = "Invalid month";
                     break;
        }
        System.out.println(monthString);
}
```

Here’s how it works:

1. Compiler generates a jump table for switch case statement
2. The switch variable/expression is evaluated once
3. Switch statement looks up the evaluated variable/expression in the jump table and directly decides which code block to execute.
4. If no match is found, then the code under default case is executed

In the above example, depending on the value of variable month , a different message will be displayed in the standard output. In this case, since the month=8, ‘August’ will be printed in standard output.

Switch statements

Although popular languages like Java and PHP have in-built switch statement, you may be surprised to know that Python language doesn’t have one. As such, you may be tempted to use a series of if-else-if blocks, using an if condition for each case of your switch statement.

However, because of the jump table, a switch statement is much faster than an if-else-if ladder. Instead of evaluating each condition sequentially, it only has to look up the evaluated variable/expression once and directly jump to the appropriate branch of code to execute it.

SEE MORE: Python jumps past Java, Javascript is still most popular language for GitHubbers (https://jaxenter.com/github-state-octoverse-2017-138103.html)

How to implement switch statement in Python

The Pythonic way to implement switch statement is to use the powerful dictionary mappings, also known as associative arrays, that provide simple one-to-one key-value mappings.

Here's the Python implementation of the above switch statement. In the following example, we create a dictionary named switcher to store all the switch-like cases.

```
def switch_demo(argument):
    switcher = {
        1: "January",
        2: "February",
        3: "March",
        4: "April",
        5: "May",
        6: "June",
        7: "July",
        8: "August",
        9: "September",
        10: "October",
        11: "November",
        12: "December"
    }
    print switcher.get(argument, "Invalid month")
```

In the above example, when you pass an argument to the switch_demo function, it is looked up against the switcher dictionary mapping. If a match is found, the associated value is printed, else a default string ('Invalid Month') is printed. The default string helps implement the 'default case' of a switch statement.

Dictionary mapping for functions

Here's where it gets more interesting. The values of a Python dictionary can be of any data type. So you don't have to confine yourself to using constants (integers, strings), you can also use function names and lambdas as values.

For example, you can also implement the above switch statement by creating a dictionary of function names as values. In this case, switcher is a dictionary of function names, and not strings.

```
def one():
    return "January"

def two():
    return "February"

def three():
    return "March"

def four():
    return "April"

def five():
    return "May"

def six():
    return "June"

def seven():
    return "July"

def eight():
    return "August"

def nine():
    return "September"

def ten():
    return "October"

def eleven():
    return "November"

def twelve():
    return "December"


def numbers_to_months(argument):
    switcher = {
        1: one,
        2: two,
        3: three,
        4: four,
        5: five,
        6: six,
        7: seven,
        8: eight,
        9: nine,
        10: ten,
        11: eleven,
        12: twelve
    }
    # Get the function from switcher dictionary
    func = switcher.get(argument, lambda: "Invalid month")
    # Execute the function
    print func()
```

Although the above functions are quite simple and only return strings, you can use this approach to execute elaborate blocks of code within each function.

SEE MORE: Python's growth comes from the enormous expansion of data science and machine learning (https://jaxenter.com/python-robinson-interview-137474.html)

In fact, if you're calling methods on objects, you can even use a dispatch method to dynamically determine which function needs to be called during runtime.

```
class Switcher(object):
    def numbers_to_months(self, argument):
        """Dispatch method"""
        method_name = 'month_' + str(argument)
        # Get the method from 'self'. Default to a lambda.
        method = getattr(self, method_name, lambda: "Invalid month")
        # Call the method as we return it
        return method()

    def month_1(self):
        return "January"

    def month_2(self):
        return "February"

    def month_3(self):
        return "March"

...
```

Based on the passed argument, the in-built getattr() function will retrieve object methods with the particular name.

```
Input: a=Switcher()
Input: a.numbers_to_months(1)
Output: January
```

Advantage of Python's approach

Since you can alter Python dictionaries during runtime (add, remove or update key-value pairs), you can easily change your very switch statement on the fly. Here's an example,

```
def zero():
    return "zero"

def one():
    return "one"

def two():
    return "two"

switcher = {
        0: zero,
        1: one,
        2: two
    }


def numbers_to_strings(argument):
    # Get the function from switcher dictionary
    func = switcher.get(argument, "nothing")
    # Execute the function
    return func()

Input: numbers_to_strings(1)
Output: One

Input: switcher[1]=two #changing the switch case
Input: numbers_to_strings(1)
Output: Two
```

Switch case is a very useful programming construct that not only provides better performance than an if-else statement but also leaves you with a more manageable code. If you've felt limited by the lack of switch statement in Python, then hopefully, the above tutorial will help you implement it.

Be the first to share this article with your network!

reddit (http://www.reddit.com/submit?url=https://jaxenter.com/implement-switch-case-statement-

Twitter (https://twitter.com/intent/tweet?status=How to implement a switch-case statement in Python https://jaxenter.com/implement-switch-case-statement-python-138315.html)

LinkedIn (https://www.linkedin.com/shareArticle?mini=true&url=https://jaxenter.com/implement-switch-case-statement-python-138315.html)

Facebook (https://www.facebook.com/sharer/sharer.php?u=https://jaxenter.com/implement-switch-case-statement-python-138315.html)

Google+ (https://plus.google.com/share?url=https://jaxenter.com/implement-switch-case-statement-python-138315.html)

Author

Sreeram Sreenivasan

For more than 8 years, Sreeram Sreenivasan has worked with various Fortune 500 Companies in areas of Business Intelligence, Sales & Marketing Strategy. He regularly writes at Fedingo about a wide range of business growth & marketing topics. He's also the Founder & CEO of Ubiq BI (https://ubiq.co/bi-solution), a cloud-based BI Platform for SMBs & Enterprises.

Recommended For You

How to set up a continuous integration process in the cloud (https://jaxenter.com/set-continuous-integration-process-cloud-140328.html)

Leave a Reply

2 Comments on "How to implement a switch-case statement in Python"

Join the discussion

Subscribe

Elena Lauren (https://mindmajix.com/python-training)

Guest

Your Article is very Interesting

Reply

3 months ago

Daniel Lauck

Guest

Nice!

Reply

1 month ago

(https://jaxenter.com/)

SEARCH

MEET US IN LONDON!

DESIGNING A HIGH-PERFORMANCE BLOCKCHAIN IN JAVA

Peter Lawrey (*Chronicle Software*)

STREAM PROCESSING WITH APACHE KAFKA AND KSQL

Tim Berglund (*Confluent*)

DEBUG EVERYTHING IN THE CLOUD WITH ECLIPSE CHE

Tracy Miranda (*Kichwa Coders*)

CLOUD NATIVE JAVA WITH OPENJ9: FAST, LEAN AND DEFINITELY MEAN

Steve Poole (*IBM*)

DEVOPS WITH KUBERNETES AND HELM

Jessica Deen (*Microsoft*)

CLOUD NATIVE APPLICATIONS MADE EASY — DON'T CODE. GENERATE!

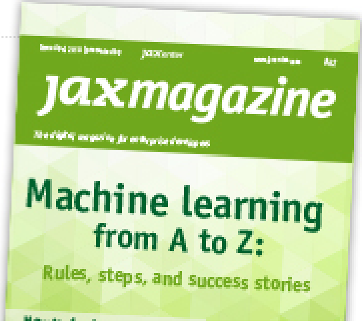
Bethany Griggs (*IBM*)

JAX London Program

PDF MAGAZINE

Machine learning in practice: Rules, steps, and success stories

free download



(http://jaxenter.com/jax-magazine)

(https://jaxenter.com/)

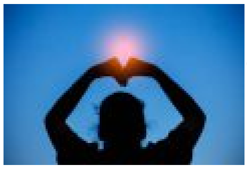
SEARCH

FEATURED POSTS

On the road to Angular v7: 6.1.0 beta.1 is here (https://jaxenter.com/road-to-angular-v7-145326.html)

Kotlin 1.2.50 is a sight for sore eyes (https://jaxenter.com/road-to-kotlin-1-2-140583.html)

If you like Vue.js, you will feel right at home with Vue Native (https://jaxenter.com/vue-js-vue-native-mobile-145571.html)



“We need to increase the awareness of the benefits and challenges of diversity” (https://jaxenter.com/diversity-talk-ix-chel-ruiz-145475.html)

JAX DEVELOPER PUZZLE

(https://jaxenter.com/jax-developers-puzzle-145121.html)

(https://jaxenter.com/)

SEARCH

Tweets by @JAXenterCOM

JAXenter.com
@JAXenterCOM
Latest and greatest: #Elasticsearch 6.3.0 features #Java10 support buff.ly/2sWUALt

Embed View on Twitter

TRENDING POSTS

How to (https://jaxenter.com/implement-switch-case-statement-python-138315.html) implement a switch-case statement in Python (https://jaxenter.com/implement-switch-case-statement-python-138315.html)

(https://jaxenter.com/10-sql-tricks-that-you-didnt-think-were-possible-125934.html) 10 SQL tricks that you didn’t think were possible (https://jaxenter.com/10-sql-tricks-that-you-didnt-think-were-possible-125934.html)

(https://jaxenter.com/)

SEARCH

(https://jaxenter.com/spring-boot-tutorial-rest-services-and-microservices-135148.html)

Spring Boot tutorial: REST services and microservices (https://jaxenter.com/spring-boot-tutorial-rest-services-and-microservices-135148.html)

Java 10 after 9 is fine; Java 10 after 8 (or earlier) be…

(https://jaxenter.com/java-10-interview-series-part-4-143144.html)

(https://jaxenter.com/java-10-interview-series-part-4-143144.html)

TIPS, TRICKS AND TUTORIALS

(http://jaxenter.com/tag/tutorial)

A little birdie told me: JAX London super early bird special ends today! (https://jaxenter.com/jax-london-super-early-bird-145323.html)

(https://jaxenter.com/jax-london-super-early-bird-145323.html)

Keystone – An OpenStack Identity Service tutorial (https://jaxenter.com/keystone-openstack-identity-service-143274.html)

(https://jaxenter.com/keystone-openstack-identity-service-143274.html)

TOPICS

DevOps (https://jaxenter.com/tag/devops)

Agile (https://jaxenter.com/tag/agile-2)

Java (https://jaxenter.com/tag/java-2)

Careers (https://jaxenter.com/tag/careers)

Open Source (https://jaxenter.com/tag/open-source)

IoT (https://jaxenter.com/tag/iot)

(https://jaxenter.com/)

SEARCH

NetBeans (/netbeans)

Eclipse (https://jaxenter.com/tag/eclipse-2)

JavaScript (https://jaxenter.com/tag/javascript)

Blockchain (https://jaxenter.com/tag/blockchain)

Tutorials (https://jaxenter.com/tag/tutorial)

PAGES

Contact (https://jaxenter.com/contact)

Newsletter (https://jaxenter.com/newsletter)

Authors (https://jaxenter.com/authors)

Found a bug? (https://jaxenter.com/found-bug)

Advertise (https://jaxenter.com/advertise)

Privacy Policy (https://jaxenter.com/privacy-policy)

Terms of Use (https://jaxenter.com/terms)

Imprint (https://jaxenter.com/imprint)

FOLLOW JAXENTER

Twitter (https://twitter.com/jaxentercom)

Facebook (https://www.facebook.com/pages/JAXentercom/123294781032857?fref=ts)

Google+ (https://plus.google.com/108137535157006873567/)

RSS (http://jaxenter.com/rss)

S&S MEDIA

JAXenter.de (http://jaxenter.de/)

JAX Finance (http://jax-finance.com/)

JAX London (https://jaxlondon.com/)

JAX Germany (https://jax.de/)

DevOpsCon (http://devopsconference.de/en/)

Developer.Press (http://developerpress.com/)

International PHP Conference (https://phpconference.com/2014/en)

Webinale (https://webinale.de/2015/)

WebMagazin (https://webmagazin.de/english)

S&S Media (http://sandsmedia.com/en)

Software & Support Media Group (http://sandsmedia.com) | Contact (https://jaxenter.com/contact)