aws re/start

Functions

Python Fundamentals

PYTHNPROG_FUNCTIONS_300_EN

Name of presenter

Date

Welcome to Functions.

What you will learn

At the core of the lesson

You will learn how to:

- Explain the purpose of functions
- Name the different types of functions
- Use functions to organize Python code

aws re/start

In this module, you will learn how to:

- Explain the purpose of functions
- Name the different types of functions
- Use functions to organize Python code

Functions

In Python, a function is a *named sequence of statements that belong together*.

Their primary purpose is to help organize programs into chunks that match how you think about the solution to the problem.

First, define the function. Name it and put placeholders for arguments. Indent lines of code inside the function, like code in loops.

Example:

```
def <function name>(argument):
    <things to do>
```

Functions, continued

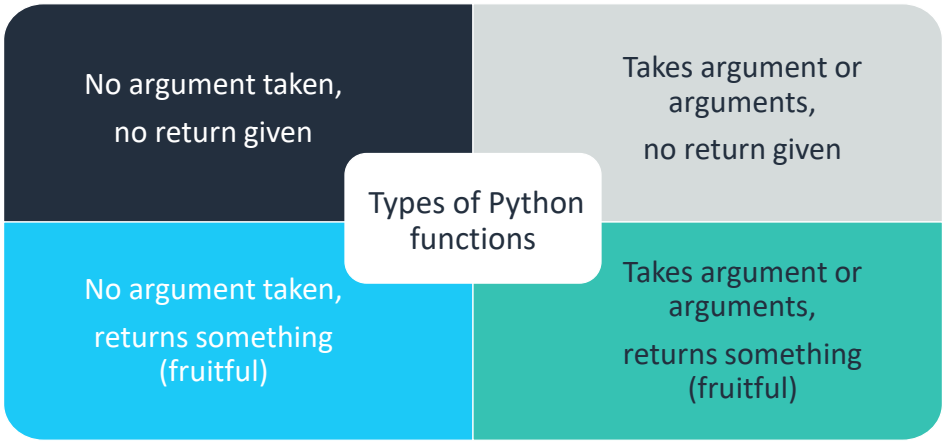
Functions are used when you must use the same block of code several times. Reusability is the primary reason for functions.

Built-in functions (like **print**) are part of a programming language, or developers can create new functions.

Python has many built-in functions that solve many common problems for you.

Examples:
- `print()`
- `open()`
- `sum()`
- `dict()`
- and others

Types of functions



Example function 1

```
def demo(x):
        y = x + 3
        return y

print(demo(3))
```

- What is the identifier (name of this function)?
- What is the argument?
- What is it returning?
- Is this function fruitful or non-fruitful?

On the print line, multiple things are happening:

- The function **demo** is being called. Functions do not run until you call them.
- The **print** function is also being called, and it uses ***demo*** as the argument for `print()`. It can be helpful to call a function with another function.

- What output do you expect to be printed?

Example function 2

```
a = 3
b = 2
c = 1

def demo():
        y =(a+b+c)

demo
```

- Is this function fruitful or non-fruitful?
- When you run this function, what do you expect to happen?
- What might be some of the advantages of using an argument in a function?

Organizing code with functions

Organizing code with functions makes it easier to read.

Example:

- It can be difficult to interpret what the first line of code does.
- Using an appropriately named function that takes appropriately named arguments makes the code easier to interpret and use.

```
#defines the value of pi
pi = 3.14159

#Calculates the area of a circle for a given radius
def calculate_circle_area(radius):
    #pi multiplied by r squared
    return pi*radius**2

r = int(input('Enter the radius of the circle: '))
area = calculate_circle_area(r)
print(area)
```

circle.py - Stoppe

Run Comma circle.py

```
Enter the radius of the circle: 5
78.53975
```


aws re/start

The **input** function prompts a message and waits for the user to enter a value (5 in this example)
This value is converted into an integer by the **int(…)** function and stored in the variable **r**
The function **calculate_area_circle** is called with **r** as a parameter. It returns the area of a circle for a radius of value **r**
The value returned by **calculate_area_circle** is stored in the variable **area**
The **print** function displays the value of the variable **area** in the console

Demonstration: Use Functions to Organize and Reuse Code

1. Write a simple function.
2. Run a sample program.
3. Review the results.
4. Define and use functions without arguments.
5. Define and use functions with arguments.
6. Compare outputs.

aws re/start

Sample code:
Items 1, 2, 3, & 4

```
def greet_user():
    print("Hello there!")


greet_user()
```

Item 5

```
def greet_user(name):
    print("Hello " + name)

greet_user("Sam")
```

Checkpoint questions

- What are arguments?
- True or False: All functions must return a value.
- How do you call a function?
- Why do you use functions?
- Name one built-in function in Python.

aws re/start

Answers:

1. Function arguments enable developers to pass values to a function. For example, a function that is called *setColor* could include a string for the color that is being passed. Such a call would appear as *setColor("red")*.
2. False.
3. You use the name of the function and then its argument in parentheses (which might be empty). For example: *setColor("red")*, ***clearScreen()***
4. Functions enable developers to use the same code many times without retyping the statements.
5. One of the most commonly used built-in functions is print.

Key takeaways

- Functions are used when you must perform the same task multiple times in a program.
- Functions are called by name and the function call often includes arguments that the function code needs for processing.
- Python includes many built-in functions, such as **print** and **help**.

aws re/start

Some key takeaways from this lesson include:

- Functions are used when you must perform the same task multiple times in a program.
- Functions are called by name and often include arguments that the code in the function needs for processing.
- Python includes many built-in functions such as print and help.