

Parameters

Creative Computing
2010 - 02 - 10

Syntax

- Structural elements of a program
- Parentheses after function names
- Colon when defining functions
- Indentation
- Lots of it to get familiar with!

Order of execution

- Python starts at the top
- Skips all definitions
- Executes commands sequentially, “jumping” into functions

Style

- Group all functions at top
- Code that Python executes is at bottom
- Use as many functions as you can
- Aim for human readability
- Use comments on tricky bits

Redundancy

- Core topic in programming
- Make the computer do the work!
- Functions were a good first step for reusing blocks

Parameters

- Information we give to a function
- Function is then reusable in different contexts

```
def function_name(<parameter_name>):  
    statement  
    statement
```

Parameters

- When Python hits a function call with a parameter, it replaces parameter name with value

```
def square(size):  
    for i in range(4):  
        forward(size)  
        right(90)
```

```
forward(100)  
square(10)
```

Parameters

- When Python hits a function call with a parameter, it replaces parameter name with value

```
def forward(distance):  
    ...  
    ...
```

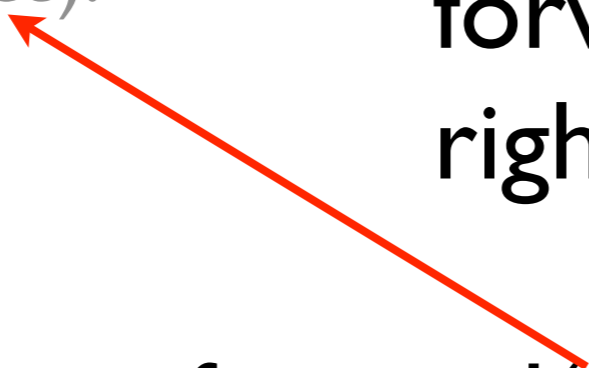
```
def square(size):  
    for i in range(4):  
        forward(size)  
        right(90)
```

```
forward(100)  
square(10)
```

Parameters

- When Python hits a function call with a parameter, it replaces parameter name with value

```
def forward(distance):  
    ...  
    ...  
  
def square(size):  
    for i in range(4):  
        forward(size)  
        right(90)  
  
forward(100)  
square(10)
```



Parameters

- When Python hits a function call with a parameter, it replaces parameter name with value

```
def forward(distance):  
    ...  
    ...  
  
def square(size):  
    for i in range(4):  
        forward(size)  
        right(90)  
  
forward(100)  
square(10)
```

The diagram illustrates the process of parameter passing. A red arrow originates from the parameter 'size' in the function call 'square(10)' and points to the parameter 'size' in the function definition 'def square(size):'. Another red arrow originates from the parameter 'distance' in the function call 'forward(100)' and points to the parameter 'distance' in the function definition 'def forward(distance):'. A third red arrow originates from the parameter 'size' in the function call 'square(10)' and points to the parameter 'size' in the function definition 'def square(size):'.

Parameters

- When Python hits a function call with a parameter, it replaces parameter name with value

```
def forward(distance):  
    ...  
    ...  
  
def square(size):  
    for i in range(4):  
        forward(size)  
        right(90)  
  
forward(100)  
square(10)
```

The diagram illustrates the flow of a parameter value. A red '10' is positioned above the 'size' parameter in the `square` function definition, with a red diagonal line striking through it. A red arrow points from this '10' to the `size` parameter in the `square` function definition. Another red arrow points from the `size` parameter in the `square` function definition to the `forward` function call `forward(size)` inside the `square` function. A third red arrow points from the `square(10)` call at the bottom to the `forward(100)` call above it. A fourth red arrow points from the `square(10)` call to the `size` parameter in the `square` function definition.

Parameters

- When Python hits a function call with a parameter, it replaces parameter name with value

```
def forward(distance):  
    ...  
    ...  
  
def square(size):  
    for i in range(4):  
        forward(size) 10  
        right(90)  
  
forward(100)  
square(10)
```

Masking

- Be careful of using the same name multiple times!
- For example, can't use color as a parameter name if you want to also use the color function
- In this case, the color parameter takes over