

V.FN.3P

Calling functions in Python

Calling a Python function with an Argument

- Each input is referred to as an **argument**
- An argument can be
 - A literal value (e.g. 4, or “hello”)

```
num1 = 42
```

```
num2 = 31
```

```
print( "hello" )
```

Argument



Calling a Python function with an Argument

- Each input is referred to as an **argument**
- An argument can be
 - A literal value (e.g. 4, or “hello”)
 - The value of a variable

```
num1 = 42
```

```
num2 = 31
```

```
print( "hello" )
```

```
print( num1 )
```

Argument



Calling a Python function with an Argument

- Each input is referred to as an **argument**
- An argument can be
 - A literal value (e.g. 4, or “hello”)
 - The value of a variable
 - Something more complicated, such as result of an expression

```
num1 = 42
```

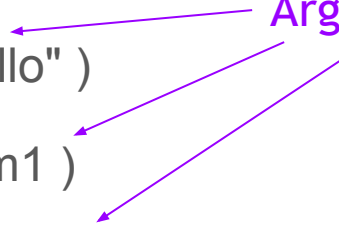
```
num2 = 31
```

```
print( "hello" )
```

```
print( num1 )
```

```
print( num1+num2 )
```

Argument



Calling a Python function with Positional Arguments

- When there is more than one argument, how does Python know what to print first?
 - It examines the arguments by position.
 - So, a more precise term for each arguments in this example is **positional argument**

```
num1 = 42
```

```
num2 = 31
```

```
print( "hello" )
```


```
print( "Nums are", num1, num2 )
```

```
print( "Their sum is", num1+num2 )
```

Positional
Argument



Positional
Argument



Calling a Python function with Keyword Arguments

- When we supply the parameter's name, it is called a **keyword argument**
- This allows us to write more readable code. We know what the purpose of each argument is
- Rule: Keyword arguments must come after all positional arguments

```
num1 = 42
```

```
num2 = 31
```

```
print( num1, num2, sep="+")
```

```
print( sep="+", num1 )
```

Keyword
Argument



Illegal: positional
argument after keyword
argument



Calling a Python function with output from another function

```
nums = [5,6,7]
```

```
N = len(nums)
```

```
print( "Printing len from variable", N )
```

```
print( "Printing len from function output", len( nums ) )
```

```
mean = sum(nums) / len(nums)
```

```
print( "Average is", mean )
```

Calls **len** and
then passes
result to **print**

