

Lecture 13, 29 September 2026

Default values for function parameters

- Can provide a default value for a parameter
- If no argument is passed, default value is used
- Recall `int(s)` converts a string `s` into an `int` if it makes sense

```
In [1]: int("7")
```

```
Out[1]: 7
```

```
In [2]: int("AB")
```

```
-----  
-  
ValueError                                Traceback (most recent call las  
t)  
Cell In[2], line 1  
----> 1 int("AB")  
  
ValueError: invalid literal for int() with base 10: 'AB'
```

- The error message above refers to `base 10`
- `int()` has an optional second parameter, which is the base
- If we specify base 16, the previous conversion works

```
In [3]: int("AB",16)
```

```
Out[3]: 171
```

- The way such a function is defined is as follows

```
def int(s,b=10):  
    ....
```

- If we call this function with only one argument, the default value of `b` is used, so the conversion is in base 10
- If we pass two arguments, the second value is used to set `b`

- Here is another example, to illustrate the point

```
In [4]: def checkdef(x = 0, y = 0):  
        print("x:",x,"y:",y)
```

```
In [5]: checkdef()
```

```
x: 0 y: 0
```

```
In [6]: checkdef(7) # 7 is used for the first parameter x, y gets the default va
x: 7 y: 0
```

- We can also pass parameters by name, ignoring the order

```
In [7]: checkdef(y=17,x=12)
x: 12 y: 17
```

- Using this, we can pass an argument for `y` and use the default `x`

```
In [8]: checkdef(y=9)
x: 0 y: 9
```