DOCSTRINGS AND ANNOTATIONS

Docstrings

We have seen the `help(x)` function before → returns some documentation (if available) for `x`

We can document our functions (and modules, classes, etc) to achieve the same result using docstrings → PEP 257

If the first line in the function body is a string (not an assignment, not a comment, just a string by itself), it will be interpreted as a docstring

```
def my_func(a):
    "documentation for my_func"
    return a
```

```
help(my_func)
```

→

```
my_func(a)
    documentation for my_func
```

Multi-line docstrings are achieved using…

multi-line strings!

Where are docstrings stored? In the function's `__doc__` property

```
def fact(n):
    """Calculates n! (factorial function)

    Inputs:
        n: non-negative integer
    Returns:
        the factorial of n
    """
    ...
```

`fact.__doc__` → 'Calculates n! (factorial function)\n \n Inputs:\n n: non-negative integer\n Returns:\n the factorial of n\n '

`help(fact)` →

```
fact(n)
    Calculates n! (factorial function)

    Inputs:
        n: non-negative integer
    Returns:
        the factorial of n
```

Function Annotations

Function annotations give us an additional way to document our functions: → PEP 3107

```
def my_func(a: <expression>, b: <expression>) -> <expression>:
    pass
```

```
def my_func(a: 'a string', b: 'a positive integer') -> 'a string':
    return a * b
```

```
help(my_func)
```
→ `my_func(a:'a string', b:'a positive integer') -> 'a string'`

```
my_func.__doc__
```
→ empty string

Annotations can be any expression

```
def my_func(a: str, b: 'int > 0') -> str:
    return a*b
```

```
def my_func(a: str, b: [1, 2, 3]) -> str:
    return a*b
```

```
x = 3
y = 5
def my_func(a: str) -> 'a repeated ' + str(max(x, y)) + ' times':
    return a*max(x, y)
```

```
help(my_func) → my_func(a:str) -> 'a repeated 5 times'
```

Default values, *args, **kwargs

can still be used as before

```
def my_func(a: str = 'xyz', b: int = 1) -> str:
    pass
```

```
def my_func(a: str = 'xyz',
            *args: 'additional parameters',
            b: int = 1,
            **kwargs: 'additional keyword only params') -> str:
    pass
```

Where are annotations stored?

In the `__annotations__` property of the function

→ dictionary

keys are the parameter names

for a return annotation, the key is `return`

values are the annotations

```
def my_func(a: 'info on a', b: int) -> float:
    pass
```

```
my_func.__annotations__
```

→ `{'a': 'info on a', 'b': int, 'return': float}`

Where does Python use docstrings and annotations?

It doesn't really!

Mainly used by external tools and modules

Example: apps that generate documentation from your code (Sphinx)

Docstrings and annotations are entirely optional, and do not "force" anything in our Python code

We'll look at an enhanced version of annotations in an upcoming section on type hints

Code