PYTHON FUNCTIONS CHEATSHEET

1. Defining a Function:

```
def function_name(parameters):
"""Docstring"""
statement(s)
return value
```

Example:
```
def greet(name):
return f"Hello, {name}!"
```

2. Function Parameters:

- Positional arguments: Passed in order.
- Keyword arguments: Passed by name.
- Default arguments: def func(x=10):
- Variable-length args:
*args - Non-keyword arguments
**kwargs - Keyword arguments

Example:
```
def add(*args):
return sum(args)

def show_info(**kwargs):
for key, value in kwargs.items():
print(f"{key}: {value}")
```

3. Lambda Functions:

```
lambda arguments: expression
```

Example:
```
square = lambda x: x**2
add = lambda a, b: a + b
```

4. Scope and Namespace:

- Local: Inside function
- Global: Outside function
- nonlocal: Modify variables in enclosing scope

Example:
```
x = 10
def modify():
global x
x = 20
```

5. Return Values:

- return exits the function and sends a value back.

Example:
def multiply(a, b):
return a * b

6. Type Hints:
---
def greet(name: str) -> str:
return f"Hello {name}"

7. Higher-Order Functions:

Functions that take other functions as arguments or return functions.

Example:
def apply(func, value):
return func(value)

8. Decorators:

Functions that modify other functions.

Example:
def decorator(func):
def wrapper():
print("Before")
func()
print("After")
return wrapper

@decorator
def say_hello():
print("Hello!")

9. Useful Built-in Functions for Functions:

- map(func, iterable)
- filter(func, iterable)
- reduce(func, iterable) # from functools import reduce
- zip(iter1, iter2)
- sorted(iterable, key=func)

10. Docstrings and Help:

```
def func():
"""This is a docstring"""
help(func)
```