PYTHON CHEATSHEET

A

- **Array Module**: For working with arrays.

python Copy Edit

```python
import array
arr = array.array('i', [1, 2, 3])
arr.append(4)
```

- **all()**: Returns `True` if all elements in an iterable are true.

python Copy Edit

```python
all([True, True, False])  # False
```

- **any()**: Returns `True` if any element in an iterable is true.

python Copy Edit

```python
any([True, False, False])  # True
```

B

- **Backslash Escape Sequences**: Used for special characters in strings.

python Copy Edit

```python
print("Hello\nWorld")  # Prints "Hello" on one line and "World" on the next
print("C:\\Users\\Name")  # Escapes backslashes
```

- **Binary**: Convert numbers to binary.

python Copy Edit

```python
bin(5)  # '0b101'
```

C

- **Counters**: From `collections` module, used for counting items.

python Copy Edit

```python
from collections import Counter
counter = Counter([1, 2, 2, 3, 3, 3])
print(counter)  # Counter({3: 3, 2: 2, 1: 1})
```

- **Copying**: Creating shallow and deep copies of objects.

python Copy Edit

```python
import copy
shallow_copy = my_list.copy()
deep_copy = copy.deepcopy(my_list)
```

D

- **Del**: Deletes an object or slice from a list.

python Copy Edit

```python
del my_list[0]  # Removes the first element
```

- **Decorators**: Functions that modify other functions.

python Copy Edit

```python
def decorator(func):
    def wrapper():
        print("Before function call")
        func()
        print("After function call")
    return wrapper

@decorator
def greet():
    print("Hello!")
```

E

- **Enum**: From `enum` module, for creating enumerations.

```python
from enum import Enum
class Color(Enum):
    RED = 1
    GREEN = 2
    BLUE = 3
```

F

- **Filter**: Filters an iterable based on a function.

```python
nums = [1, 2, 3, 4, 5]
even_nums = list(filter(lambda x: x % 2 == 0, nums))
```

- **Format()**: Used for string formatting.

```python
name = "Alice"
age = 30
print("My name is {} and I am {} years old.".format(name, age))
```

G

- **Global and Local Scope**: Variables defined inside a function are local, while those outside are global.

```python
global_var = "I am global"
def test():
    local_var = "I am local"
    print(global_var)
```

- **Get()**: Used with dictionaries to retrieve values with a default.

python Copy Edit

```python
my_dict = {"name": "John"}
print(my_dict.get("age", "Not available"))  # "Not available"
```

H

- **Hasattr()**: Checks if an object has a particular attribute.

python Copy Edit

```python
class Person:
    name = "John"
p = Person()
print(hasattr(p, "name"))  # True
```

I

- **In and Not In**: Checks membership in iterables.

python Copy Edit

```python
3 in [1, 2, 3]  # True
4 not in [1, 2, 3]  # True
```

- **Isinstance()**: Checks if an object is an instance of a class.

python Copy Edit

```python
isinstance(5, int)  # True
isinstance("Hello", str)  # True
```

J

- **JSON**: Work with JSON data.

```python
import json
data = '{"name": "John", "age": 30}'
parsed_data = json.loads(data)
print(parsed_data["name"])  # John
```

L

- **Length of List**: Use `len()` to find the number of elements in a list.

```python
len([1, 2, 3])  # 3
```

- **Local Variables**: Variables defined inside a function.

```python
def test():
    x = 10  # Local variable
```

M

- **Max()**: Returns the largest item.

```python
max([1, 2, 3, 4])  # 4
```

- **Min()**: Returns the smallest item.

```python
min([1, 2, 3, 4])  # 1
```

N

- **Next()**: Returns the next item from an iterator.

```python
nums = iter([1, 2, 3])
print(next(nums))  # 1
```

O

- **Open()**: Opens a file for reading or writing.

```python
with open('file.txt', 'r') as file:
    data = file.read()
```

- **Or**: Logical OR operation.

```python
True or False  # True
```

P

- **Pass**: A placeholder for future code.

```python
def func():
    pass  # No implementation yet
```

- **Print()**: Prints values to the console.

```python
print("Hello, World!")
```

- **Pop()**: Removes an item from a list and returns it.

```python
my_list = [1, 2, 3]
item = my_list.pop(1)  # Removes item at index 1 (value 2)
```

Q

- **Queue**: From the `queue` module, for thread-safe queues.

```python
from queue import Queue
q = Queue()
q.put(1)
q.put(2)
print(q.get())  # 1
```

R

- **Reversed()**: Returns the reversed version of an iterable.

```python
reversed_list = list(reversed([1, 2, 3]))
```

- **Round()**: Rounds a number to a specified decimal place.

```python
round(3.14159, 2)  # 3.14
```

S

- **Strip()**: Removes leading and trailing whitespace from a string.

```python
"  hello  ".strip()  # "hello"
```

- **Set Comprehensions**: Create sets using a similar syntax to list comprehensions.

```python
even_numbers = {x for x in range(10) if x % 2 == 0}
```

- **Sort()**: Sorts a list in-place.

```python
my_list = [3, 1, 2]
my_list.sort()  # [1, 2, 3]
```

T

- **Try-Else**: Executes code when no exception occurs in `try`.

```python
try:
    x = 1 / 1
except ZeroDivisionError:
    print("Error")
else:
    print("No error")
```

- **Tuple Unpacking**: Assign elements of a tuple to variables.

```python
a, b = (1, 2)
```

U

- **Unpacking**: Unpack iterables into variables.

```python
a, b, *rest = [1, 2, 3, 4, 5]
```

- **Upper()**: Converts a string to uppercase.

```python
"hello".upper()  # "HELLO"
```

V

- **Varargs**: Variable length arguments.

```python
def sum_all(*args):
    return sum(args)
```

- **Zip()**: Combine multiple iterables into a single iterable.

```python
zip([1, 2], ['a', 'b'])  # [(1, 'a'), (2, 'b')]
```

W

- **While-Else**: Executes code when the while loop ends normally.

```python
x = 0
while x < 5:
    print(x)
    x += 1
else:
    print("Loop finished")
```

- **With Statement**: For managing resources, like file handling.

```python
with open("file.txt", "w") as f:
    f.write("Hello!")
```

X

- **Xrange()**: Used in Python 2 for generating numbers lazily. (Python 3 uses `range()` which is like `xrange()`.)

Y

- **Yield From**: Used to delegate part of a generator to another generator.

```python
def generator():
    yield from range(5)
```

Z

- **Zero Division Error**: Caught by `except` in try-except blocks.

```python
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

This complete cheat sheet should cover most of the Python concepts and functions you may encounter!