CodeWithHarry

PYTHON

CHEATSHEET

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

# Variables
x = 10            # int
y = 3.14          # float
name = "Harry"    # str
is_active = True  # bool

# If-Else
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")

# For Loop
for i in range(5):
    print(i)

# Function
def greet(name):
    return f"Hello, {name}!"
print(greet(name))
```

BASIC SYNTAX
Indentation, Statements, Comments, Input/Output

DATA TYPES
Numbers, String, List, Tuple, Set, Dictionary

OPERATORS
Arithmetic, Comparison, Logical, Assignment

CONTROL FLOW
if, elif, else, for, while, break, continue

FUNCTIONS
def, return, *args, **kwargs, lambda

COLLECTIONS
List, Tuple, Set, Dictionary

MODULES & PACKAGES
import, from...import, as, pip

EXCEPTION HANDLING
try, except, else, finally, raise

BASIC SYNTAX | DATA TYPES | CONTROL FLOW | FUNCTIONS | MODULES | EXCEPTIONS | FILE HANDLING

Python Programming

Introduction & Installation

Cheatsheet by CodeWithHarry

Installation

WEB

Go to python.org/downloads and install the latest version

Check Python Version

```
python --version
python3 --version
py --version
```

Run Python

```
python main.py
```

Basic Hello World Syntax

PYTHON FILE

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

Python Programming

Cheatsheet by CodeWithHarry

Variables & Data Types

Variables

STRING

```
name = "Harry"
```

INTEGER

```
age = 25
```

FLOAT

```
price = 99.99
```

BOOLEAN

```
is_active = True
```

MULTIPLE

```
x, y = 10, 20
```

SAME VALUE

```
a = b = c = 0
```

Data Types

TEXT

`str`

NUMBER

`int, float, complex`

SEQUENCE

`list, tuple, range`

MAPPING

`dict`

SET

`set, frozenset`

BOOLEAN

`bool`

Type Checking

PYTHON

```python
x = 5
print(type(x))
print(isinstance(x, int))
```

Constant Naming Convention

Constant names are typically written in uppercase letters. This is a convention, not enforced!

```
PI = 3.14
```

```
MAX_SIZE = 100
```

Python Programming

Cheatsheet by CodeWithHarry

Operators & Type Conversion

Operators

ARITHMETIC
`+, -, *, /, %, //, **`

ASSIGNMENT
`=, +=, -=, *=, /=, %=`

COMPARISON
`==, !=, >, <, >=, <=`

LOGICAL
`and, or, not`

MEMBERSHIP
`in, not in`

IDENTITY
`is, is not`

Operator Precedence

1. PARENTHESES
`()` first

2. POWER
`**`

3. MULTIPLY
`*, /, //, %`

4. ADD
`+, -`

5. COMPARE
`==, !=, <, >, in, is`

6. NOT
`not`

7. AND
`and`

8. OR
`or`

Operator Syntax

ADD
```
x + y
```

FLOOR DIV
```
x // y
```

POWER
```
x ** y
```

EQUAL
```
x == y
```

LOGIC
```
x > 10 and x < 20
```

MEMBER
```
"a" in "harry"
```

IDENTITY
```
x is None
```

Type Conversion

INTEGER (immutable)
```
int("10")
```

FLOAT (immutable)
```
float("10.5")
```

STRING (immutable)
```
str(100)
```

BOOLEAN (immutable)
`bool(0)` is False;
`bool("False")` is True

LIST (mutable)
```
list("abc")
```

TUPLE (immutable)
```
tuple([1, 2])
```

Python Programming

Strings

Cheatsheet by CodeWithHarry

Create Strings

SINGLE
```
s = 'hello'
```

DOUBLE
```
s = "hello"
```

MULTILINE
```
s = '''hello'''
```

RAW
```
r"C:\new\file"
```

Indexing & Slicing

Assume s = "Harry"

FIRST
```
s[0] -> "H"
```

LAST
```
s[-1] -> "y"
```

SLICE
```
s[1:4] -> "arr"
```

STEP
```
s[::2] -> "Hry"
```

REVERSE
```
s[::-1] -> "yrraH"
```

LENGTH
```
len(s) -> 5
```

String Methods

Assume s = "Harry"

UPPER
```
s.upper() -> "HARRY"
```

LOWER
```
s.lower() -> "harry"
```

SPACES
```
s.strip() -> "Harry"
```

REPLACE
```
s.replace("a", "o") -> "Horry"
```

SPLIT
```
s.split("r") -> ["Ha", "", "y"]
```

JOIN
```
"-".join(s) -> "H-a-r-r-y"
```

FIND
```
s.find("r") -> 2
```

Formatting

PYTHON

```python
name = "Harry"
age = 25
print(f"{name} is {age}")
```

Python Programming

Lists

Cheatsheet by CodeWithHarry

List Syntax

CREATE

```
lst = [1, 2, 3]
```

MIXED

```
lst = [1, "a", True]
```

EMPTY LIST

```
lst = []
```

LENGTH

```
len(lst)
```

Access & Slice

FIRST/LAST

```
lst[0] / lst[-1]
```

RANGE

```
lst[1:4]
```

REVERSE

```
lst[::-1]
```

List Methods

ADD ONE

```
lst.append(x) adds one element
```

ADD MANY

```
lst.extend(items)
```

INSERT

```
lst.insert(i, x)
```

REMOVE

```
lst.remove(x) removes first match
```

POP

```
lst.pop()
```

CLEAR

```
lst.clear()
```

SORT

```
lst.sort()
```

REVERSE

```
lst.reverse()
```

Comprehension

PYTHON

```python
squares = [x*x for x in range(5)]
evens = [x for x in nums if x % 2 == 0]
```

Python Programming

Cheatsheet by CodeWithHarry

Tuples, Sets & Dictionaries

Tuples

CREATE

```
t = (1, 2, 3)
```

ONE ITEM

```
t = (1,)
```

ACCESS

```
t[0], t[-1]
```

UNPACK

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

Sets

CREATE

```
s = {1, 2, 3}
```

EMPTY SET

```
s = set()
```

ADD

```
s.add(4)
```

REMOVE

```
s.remove(2) # error if
not present
s.discard(2) # safer
```

UNION

```
a | b
```

INTERSECTION

```
a & b
```

Dictionaries

CREATE

```
d = {"name": "Harry"}
```

ACCESS

```
d["name"]
```

GET

```
d.get("age")
```

UPDATE

```
d["age"] = 25
```

KEYS

```
d.keys()
```

ITEMS

```
d.items()
```

Python Programming

Cheatsheet by CodeWithHarry

Conditional Statements

if / elif / else

PYTHON

```python
if condition:
    statement
elif another_condition:
    statement
else:
    statement
```

Comparisons

EQUAL/ NOT EQUAL

```
x == y
x != y
```

GREATER/LESS

```
x > y
x < y
```

GTE/LTE

```
x >= y
x <= y
```

Logical Conditions

AND/OR

```
x > 0 and x < 10
x == 0 or x == 1
```

NOT

```
not is_active
```

TERNARY

```
a if condition else b
```

Match Case

PYTHON 3.10+

```python
match value:
    case 1:
        print("one")
    case _:
        print("other")
```

Python Programming

Cheatsheet by CodeWithHarry

Loops

for Loop

PSEUDO CODE

```
for item in iterable:
    statement
```

PYTHON

```python
for i in range(5):
    print(i)
```

while Loop

PSEUDO CODE

```
while condition:
    statement
```

PYTHON

```python
i = 0
while i < 5:
    i += 1
```

Loop Tools

RANGE

`range(start, stop, step)`

BREAK

`break`

CONTINUE

`continue`

PASS

`pass`

ENUMERATE

`enumerate(items)`

ZIP

`zip(a, b)`

Loop else

PYTHON

```python
for x in nums:
    if x == target:
        break
else:
    print("not found")
```

Python Programming

Functions

Cheatsheet by CodeWithHarry

Function Syntax

PYTHON

```python
def function_name(parameters):
    return value

result = function_name(arguments)
```

Parameters

NORMAL

```
def add(a, b):
```

DEFAULT

```
def greet(name="User"):
```

KEYWORD

```
greet(name="Harry")
```

ARGS

```
def f(*args):
```

KWARGS

```
def f(**kwargs):
```

ANNOTATION

```
def add(a: int) -> int:
```

Lambda

PYTHON

```python
square = lambda x: x * x
print(square(5))
```

Scope

LOCAL

```
x inside function
```

GLOBAL

```
global x
```

Python Programming

Cheatsheet by CodeWithHarry

Exception Handling

try / except

PYTHON

```python
try:
    risky_code()
except Exception as e:
    print(e)
```

Full Block

PYTHON

```python
try:
    code
except ValueError as e:
    print(e)
else:
    print("no error")
finally:
    print("always runs")
```

Common Exceptions

VALUE ValueError	TYPE TypeError	INDEX IndexError
KEY KeyError	FILE FileNotFoundError	ZERO DIV ZeroDivisionError

Raise

PYTHON

```python
if age < 0:
    raise ValueError("Invalid age")
```

Python Programming

Cheatsheet by CodeWithHarry

File Handling

Open File

READ

```
open("file.txt", "r")
```

WRITE

```
open("file.txt", "w")
```

APPEND

```
open("file.txt", "a")
```

BINARY

```
open("file.bin", "rb")
```

with Syntax

PYTHON

```python
with open("data.txt", "r") as f:
    content = f.read()
```

Read / Write

READ ALL

```
f.read()
```

READ LINE

```
f.readline()
```

READ LINES

```
f.readlines()
```

WRITE

```
f.write("hello")
```

Python Programming

Cheatsheet by CodeWithHarry

Object Oriented Programming

Class Syntax

PYTHON

```python
class Person:
    def __init__(self, name):
        self.name = name

    def greet(self):
        print(self.name)
```

Object Syntax

CREATE

```
p = Person("Harry")
```

ATTRIBUTE

```
p.name
```

METHOD

```
p.greet()
```

Inheritance

PYTHON

```python
class Student(Person):
    def __init__(self, name, marks):
        super().__init__(name)
        self.marks = marks
```

Special Methods

INIT

```
__init__(self)
```

STR

```
__str__(self)
```

LEN

```
__len__(self)
```

REPR

```
__repr__(self)
```

Python Programming

Modules & pip

Cheatsheet by CodeWithHarry

Import Syntax

MODULE

```
import math
```

ALIAS

```
import numpy as
np
```

FROM

```
from math import
sqrt
```

ALL

```
from module
import *
```

Module Usage

PYTHON

```python
import math
print(math.sqrt(16))

from random import randint
print(randint(1, 10))
```

pip Commands

INSTALL

```
pip install package
```

UNINSTALL

```
pip uninstall package
```

LIST

```
pip list
```

FREEZE

```
pip freeze
```

UPGRADE

```
pip install -U package
```

REQUIREMENTS

```
pip install -r
requirements.txt
```

Virtual Environment

TERMINAL

```
python -m venv .venv
.venv\Scripts\activate
source .venv/bin/activate
```

Python Programming

Cheatsheet by CodeWithHarry

Useful Built-in Functions

Data Functions

TYPE
`type(x)`

LENGTH
`len(x)`

INPUT
`input("Name: ")`

OUTPUT
`print(x)`

CONVERT
`int(), float(), str()`

BOOLEAN
`bool(x)`

Math Functions

SUM
`sum(nums)`

MIN
`min(nums)`

MAX
`max(nums)`

ROUND
`round(x, 2)`

ABS
`abs(x)`

POWER
`pow(x, y)`

Iterable Functions

RANGE
`range(5)`

ENUMERATE
`enumerate(items)`

ZIP
`zip(a, b)`

SORTED
`sorted(items)`

ANY
`any(values)`

ALL
`all(values)`

Examples

PYTHON

```python
nums = [3, 1, 2]
print(len(nums))
print(sorted(nums))
print(sum(nums))
```

Python Programming

Cheatsheet by CodeWithHarry

Useful Built-in Functions

USEFUL BUILT-IN FUNCTIONS

Python provides many built-in functions for common tasks.

print()
Display output

len()
Get length

type()
Get type

range()
Generate sequence

sum()
Sum items

min()/max()
Find min/max

sorted()
Sort items

enumerate()
Index with items

zip()
Combine iterables

map()
Apply function

filter()
Filter items

any()/all()
Test conditions

```
nums = [1, 2, 3]
print(sum(nums))
print(list(enumerate(nums)))
print(list(map(str, nums)))
```