PHP → Python Quick Reference

Essential Syntax Conversions

Basic Syntax

Feature	PHP	Python
Comments	`// Single line` `/* Multi line */`	`# Single line` `""" Multi line """`
Variables	`$variable = "value";`	`variable = "value"`

Constants	`define("CONST", "value");`
	`const CONST = "value";`
		`CONST = "value"`
		`# or`
		`CONST: Final = "value"`

Print	`echo "Hello";` `print("Hello");`	`print("Hello")`
Statement end	Semicolon required ;	No semicolon needed
Blocks	Braces { }	Indentation (4 spaces)

Variables & Types

Type	PHP	Python
String	`$str = "text";` `$str = 'text';`	`str = "text"` `str = 'text'`
Integer	`$num = 42;`	`num = 42`
Float	`$val = 3.14;`	`val = 3.14`
Boolean	`$flag = true;` `$flag = false;`	`flag = True` `flag = False`
Null	`$var = null;`	`var = None`

Type check	`is_string($var)`
	`is_int($var)`
	`is_bool($var)`
		`isinstance(var, str)`
		`isinstance(var, int)`
		`isinstance(var, bool)`

Type cast	`(int)$var`
	`(string)$var`
	`(array)$var`
		`int(var)`
		`str(var)`
		`list(var)`

String Operations

Operation	PHP	Python

Concatenation	`$str = "Hello" . " " . "World";`	`str = "Hello" + " " + "World"`
		`# or`
		`str = f"Hello {name}"`

Length	strlen($str)	len(str)
Substring	substr($str, 0, 5)	str[0:5]
Upper/Lower	strtoupper($str) strtolower($str)	str.upper() str.lower()
Trim	trim($str) ltrim($str) rtrim($str)	str.strip() str.lstrip() str.rstrip()
Replace	str_replace("old", "new", $str)	str.replace("old", "new")
Split	explode(",", $str)	str.split(",")
Join	implode(",", $arr)	",".join(list)
Find	strpos($str, "text")	str.find("text") # or "text" in str
Format	sprintf("%s %d", $s, $n)	f"{s} {n}" # or "{} {}".format(s, n)

Arrays, Lists & Dictionaries

Operation	PHP	Python
Indexed array	`$arr = [1, 2, 3];` `$arr = array(1, 2, 3);`	`arr = [1, 2, 3]`
Assoc array	`$arr = ["key" => "val"];` `$arr["key"] = "val";`	`arr = {"key": "val"}` `arr["key"] = "val"`
Access	`$arr[0];` `$arr["key"];`	`arr[0]` `arr["key"]`
Add element	`$arr[] = "new";` `array_push($arr, "new");`	`arr.append("new")`
Remove last	`array_pop($arr)`	`arr.pop()`
Remove first	`array_shift($arr)`	`arr.pop(0)`
Add to start	`array_unshift($arr, "new")`	`arr.insert(0, "new")`
Length	`count($arr)`	`len(arr)`
Check exists	`isset($arr["key"])` `array_key_exists("key", $arr)`	`"key" in arr`
Keys	`array_keys($arr)`	`list(dict.keys())` `# or arr.keys()`
Values	`array_values($arr)`	`list(dict.values())` `# or arr.values()`
Merge	`array_merge($a1, $a2)`	`a1 + a2  # lists` `{**d1, **d2}  # dicts`
Sort	`sort($arr);` `asort($arr); // assoc`	`arr.sort()` `sorted(arr)`
Filter	`array_filter($arr, $callback)`	`list(filter(func, arr))` `[x for x in arr if x>0]`
Map	`array_map($callback, $arr)`	`list(map(func, arr))` `[func(x) for x in arr]`

Control Structures

Structure	PHP	Python

If/Else	`if ($x > 0) {`
	`    // code`
	`} elseif ($x == 0) {`
	`    // code`
	`} else {`
	`    // code`
	`}`
		`if x > 0:`
		`    # code`
		`elif x == 0:`
		`    # code`
		`else:`
		`    # code`

Ternary	`$result = $x > 0 ? "yes" : "no";`	`result = "yes" if x > 0 else "no"`

Switch	switch($x) {
	    case 1:
	        break;
	    default:
	        break;
	}
		match x: # Python 3.10+
		    case 1:
		        pass
		    case _:
		        pass

For loop	for ($i=0; $i<10; $i++) {
	    // code
	}
		for i in range(10):
		    # code

Foreach	foreach ($arr as $item) {
	    // code
	}
	foreach ($arr as $k => $v) {
	    // code
	}
		for item in arr:
		    # code
		
		for k, v in dict.items():
		    # code

While	while ($x < 10) {
	    $x++;
	}
		while x < 10:
		    x += 1

Do-While	do {
	    $x++;
	} while ($x < 10);
		while True:
		    x += 1
		    if x >= 10:
		        break

Functions

Concept	PHP	Python

Basic	`function myFunc($a, $b) {`
	`  return $a + $b;`
	`}`
		`def my_func(a, b):`
		`  return a + b`

Default args	`function myFunc($a, $b=10) {`
	`  return $a + $b;`
	`}`
		`def my_func(a, b=10):`
		`  return a + b`

Variable args	`function myFunc(...$args) {`
	`  // $args is array`
	`}`
		`def my_func(*args):`
		`  # args is tuple`

Named args	`myFunc(a: 1, b: 2); // PHP 8+`	`my_func(a=1, b=2)`
Keyword args	`N/A in older PHP`	`def func(**kwargs):` `  # kwargs is dict`

Return type	`function myFunc(): int {`
	`  return 1;`
	`}`
		`def my_func() -> int:`
		`  return 1`

Anonymous	`$func = function($x) {`
	`  return $x * 2;`
	`};`
		`func = lambda x: x * 2`
		# or
		`def func(x):`
		`  return x * 2`

Arrow func	`$func = fn($x) => $x * 2;`	`func = lambda x: x * 2`

Object-Oriented Programming

Concept	PHP	Python

Class	`class MyClass {`
	`  public $prop;`
	
	`  public function method() {`
	`  return $this->prop;`
	`  }`
	`}`
		`class MyClass:`
		`  def __init__(self):`
		`  self.prop = None`
		
		`  def method(self):`
		`  return self.prop`

Constructor	`public function __construct($x) {`
	`  $this->prop = $x;`
	`}`
		`def __init__(self, x):`
		`  self.prop = x`

Instantiate	`$obj = new MyClass(10);`	`obj = MyClass(10)`

Properties	`public $prop;`
	`private $prop;`
	`protected $prop;`
		`self.prop  # public`
		`self._prop  # protected`
		`self.__prop  # private`

Static	`public static $prop;`
	`public static function method() {}`
	`self::$prop;`
	`self::method();`
		`class_var = value`
		`@staticmethod`
		`def method():`
		`  pass`
		`MyClass.class_var`

Inheritance	class Child extends Parent {
	    // code
	}
		class Child(Parent):
		    # code

Call parent	parent::__construct(); parent::method();	super().__init__() super().method()

Abstract	abstract class Base {
	    abstract function method();
	}
		from abc import ABC
		class Base(ABC):
		    @abstractmethod
		    def method(self):
		        pass

File & Error Handling

Operation	PHP	Python
Read file	`$content = file_get_contents("f.txt");`	`with open("f.txt") as f:` `  content = f.read()`
Write file	`file_put_contents("f.txt", $data);`	`with open("f.txt", "w") as f:` `  f.write(data)`
Append	`file_put_contents("f.txt", $d, FILE_APPEND);`	`with open("f.txt", "a") as f:` `  f.write(data)`
Read lines	`$lines = file("file.txt");`	`with open("f.txt") as f:` `  lines = f.readlines()`
Check exists	`file_exists("file.txt")`	`import os` `os.path.exists("file.txt")`
Delete	`unlink("file.txt");`	`import os` `os.remove("file.txt")`

Try/Catch	`try {`
	`  // code`
	`} catch (Exception $e) {`
	`  echo $e->getMessage();`
	`}`
		`try:`
		`  # code`
		`except Exception as e:`
		`  print(str(e))`

Throw	`throw new Exception("Error");`	`raise Exception("Error")`

Finally	`try {}`
	`catch {}`
	`finally {}`
		`try:`
		`  pass`
		`except:`
		`  pass`
		`finally:`
		`  pass`

Database Operations

Operation	PHP (MySQLi/PDO)	Python (MySQL/SQLite)

Connect	`$conn = new mysqli($h, $u, $p, $db);`
	
	`$pdo = new PDO("mysql:host=$h;dbname=$db", $u, $p);`
		`import mysql.connector`
		`conn = mysql.connector.connect(`
		`  host=h, user=u,`
		`  password=p, database=db`
		`)`

Query	`$result = $conn->query($sql);`
	
	`$stmt = $pdo->query($sql);`
		`cursor = conn.cursor()`
		`cursor.execute(sql)`
		`result = cursor.fetchall()`

Prepared	`$stmt = $conn->prepare("SELECT * FROM t WHERE id=?");`
	`$stmt->bind_param("i", $id);`
	`$stmt->execute();`
		`cursor.execute(`
		`  "SELECT * FROM t WHERE id=%s",`
		`  (id,)`
		`)`

Fetch row	`$row = $result->fetch_assoc();` `$row = $stmt->fetch();`	`row = cursor.fetchone()`

Fetch all	`$rows = $result->fetch_all(MYSQLI_ASSOC);` `$rows = $stmt->fetchAll();`	`rows = cursor.fetchall()`

Insert	`$stmt = $pdo->prepare("INSERT INTO t (name) VALUES (?)");`
	`$stmt->execute([$name]);`
		`cursor.execute(`
		`    "INSERT INTO t (name) VALUES (%s)",`
		`    (name,)`
		`)`
		`conn.commit()`

Last ID	`$id = $conn->insert_id;` `$id = $pdo->lastInsertId();`	`id = cursor.lastrowid`
Close	`$conn->close();` `$pdo = null;`	`cursor.close()` `conn.close()`

Common Patterns & Idioms

Pattern	PHP	Python
Check empty	`empty($var)` `!empty($var)`	`not var` `bool(var)`
Isset	`isset($var)`	`var is not None` `"key" in dict`
Null coalesce	`$x = $var ?? "default";` `$x = $var ?: "default";`	`x = var if var else "default"` `x = var or "default"`
Spread	`function f(...$args) {}` `f(...$array);`	`def f(*args): pass` `f(*list)`
Destructure	`list($a, $b) = $arr;` `[$a, $b] = $arr;`	`a, b = arr` `a, *rest = arr`
Range	`range(1, 10)`	`range(1, 11)` `# Note: Python excludes end`
In array	`in_array($needle, $haystack)`	`needle in haystack`
JSON encode	`json_encode($data)`	`import json` `json.dumps(data)`
JSON decode	`json_decode($str, true)`	`import json` `json.loads(str)`

Sleep	`sleep(5); // seconds`
	`usleep(5000000); // microsec`
		`import time`
		`time.sleep(5)  # seconds`

Date/Time	`date("Y-m-d H:i:s")` `strtotime($str)`	`from datetime import datetime` `datetime.now().strftime("%Y-%m-%d %H:%M:%S")`
Include	`require "file.php";` `include "file.php";`	`import module` `from module import func`
Exit	`die("message");` `exit(1);`	`import sys` `sys.exit(1)`

HTTP & Web Operations

Operation	PHP	Python
GET params	`$_GET["param"]`	`from flask import request` `request.args.get("param")`
POST data	`$_POST["field"]`	`request.form.get("field")` `request.json.get("field")`
Headers	`header("Content-Type: application/json");`	`from flask import Response` `Response(headers={...})`
Redirect	`header("Location: /page");` `exit;`	`from flask import redirect` `return redirect("/page")`
Session	`$_SESSION["key"] = "val";` `session_start();`	`from flask import session` `session["key"] = "val"`

Cookie	`setcookie("name", "val", time()+3600);`	`from flask import make_response` `resp.set_cookie("name", "val")`
HTTP GET	`file_get_contents($url)` `curl_exec(curl_init($url))`	`import requests` `requests.get(url)`

HTTP POST	`$ch = curl_init($url);`
	`curl_setopt($ch, CURLOPT_POST, 1);`
	`curl_setopt($ch, CURLOPT_POSTFIELDS, $data);`
		`requests.post(url, data=data)`
		`requests.post(url, json=data)`

Key Philosophical Differences

Indentation Matters: Python uses indentation (4 spaces) for blocks instead of braces.
No Variable Prefix: Python doesn't use $ for variables.
Everything is an Object: In Python, everything (including functions) is an object.
Duck Typing: Python uses duck typing - if it walks like a duck and quacks like a duck, it's a duck.
List Comprehensions: Python has powerful list comprehensions: `[x*2 for x in range(10) if x>5]`
Multiple Inheritance: Python supports multiple inheritance: `class Child(Parent1, Parent2)`
Generators: Python has generators using yield: `def gen(): yield 1`
Context Managers: Python's `with` statement handles resources automatically.
Decorators: Python uses `@decorator` syntax above functions for wrapping.