PYTHON REFERENCE SHEET

Essential syntax, keywords and built-in functions

How to use this reference sheet

Python has a very large number of functions and libraries, so there is no short list containing every possible syntax. This document brings together the fundamental language structures, operators, commonly used built-in functions and examples of popular libraries. Python keywords and explanations are presented in English.

Important rule: Python is case-sensitive (age, Age and AGE are three different names). Indentation (spaces at the beginning of a line) defines instruction blocks. Use four spaces for each indentation level.

1. Basic program structure

Syntax / keyword	Meaning	Example
# comment	Adds a note that Python ignores.	# My first program
print()	Displays information or output.	print('Hello')
input()	Gets text input from the user.	name = input('Name: ')
:	Introduces an indented code block.	if age >= 18:
Indentation	Uses spaces to define a block, usually four spaces.	print('OK')
import	Imports a module or library.	import math
from ... import ...	Imports a specific item from a module.	from math import sqrt
as	Gives a module or imported item an alias.	import numpy as np

2. Variables and data types

Syntax / type	Meaning	Example
name = value	Creates or updates a variable.	score = 15
int	Integer number.	age = 16
float	Floating-point number (decimal number).	temperature = 21.5
str	String / text.	city = 'London'
bool	Boolean value: True or False.	active = True
None	Represents the absence of a value.	result = None
list	Ordered, mutable collection.	marks = [12, 15, 9]
tuple	Ordered, immutable sequence.	point = (3, 4)
dict	Collection of key-value pairs.	student = {'name': 'Lina', 'age': 16}
set	Unordered collection of unique items.	colors = {'red', 'blue'}
type()	Returns the type of a value.	type(12)
isinstance()	Checks whether a value is an instance of a type.	isinstance(12, int)
int(), float(), str(), bool()	Convert a value to a specified type.	int('12')

3. Operators

Operator	Meaning	Example
+ - * /	Addition, subtraction, multiplication and division.	7 + 2
//	Floor division (quotient rounded down).	7 // 2 # 3
%	Remainder / modulo operator.	7 % 2 # 1
**	Exponentiation / power.	2 ** 3 # 8
== !=	Equal to / not equal to.	a == b
> < >= <=	Greater than, less than, greater than or equal to, less than or equal to.	age >= 18

and / or / not	Logical AND / OR / NOT.	age >= 18 and registered
in / not in	Membership / not membership in a sequence.	'a' in 'cat'
is / is not	Tests object identity, not equality of contents.	x is None
= += -= *= /=	Assignment / augmented assignment operators.	score += 1

4. Conditional statements

Syntax	Meaning	Example
if	Runs a block if a condition is true.	if mark >= 10:
elif	Checks another condition if previous conditions are false.	elif mark >= 8:
else	Runs the block in all other cases.	else:
match / case	Matches a value against patterns (Python 3.10+).	match choice:
assert	Checks a condition and raises an error if it is false.	assert age >= 0

Example:

```
mark = 13
if mark >= 10:
    print('Pass')
else:
    print('Keep practising')
```

5. Loops and repetition

Syntax	Meaning	Example
for	Loops over each item in a sequence.	for mark in marks:
while	Repeats while a condition is true.	while counter < 5:
range()	Generates a sequence of integers; the stop value is excluded.	range(0, 5)
break	Immediately exits the loop.	if x == 0: break
continue	Skips to the next iteration.	if x < 0: continue
pass	Does nothing; used as a placeholder.	def function(): pass
enumerate()	Provides both the index and the value while looping.	for i, value in enumerate(marks):
zip()	Iterates over several sequences in parallel.	for name, mark in zip(names, marks):

6. Functions

Syntax	Meaning	Example
def	Defines a function.	def greet():
return	Returns a result and ends the function.	return a + b
Parameter	Variable listed in a function definition.	def square(x):
Argument	Value passed when calling a function.	square(5)
Default value	Value used when an argument is omitted.	def greet(name='friend'):
*args	Collects a variable number of positional arguments.	def f(*args):
**kwargs	Collects a variable number of keyword arguments.	def f(**kwargs):
lambda	Creates a small anonymous function.	double = lambda x: x * 2
yield	Produces a value from a generator without ending the function.	yield number
global	Declares that a function uses a global variable.	global counter

nonlocal	Refers to a variable in an enclosing function.	nonlocal total

Example:

```
def add(a, b):
    result = a + b
    return result

print(add(4, 6))
```

7. Strings

Syntax / method	Meaning	Example
'...' / "..."	Defines a string.	name = 'Lina'
f'...{variable}...'	Inserts values into a formatted string.	f'Hello {name}'
.lower() / .upper()	Converts text to lowercase / uppercase.	'HELLO'.lower()
.strip()	Removes leading and trailing whitespace.	' hello '.strip()
.replace(a, b)	Replaces one substring with another.	'cat'.replace('c','h')
.split()	Splits a string into a list.	'a,b'.split(',')
.join()	Joins strings using a separator.	','.join(['a','b'])
.find()	Returns the position of a substring, or -1.	'python'.find('t')
.startswith() / .endswith()	Checks the beginning / end of a string.	name.startswith('L')
len()	Returns the number of characters.	len('Python')
[start:stop:step]	Slices a sequence; the stop index is excluded.	text[0:3]

8. Lists, tuples, dictionaries and sets

Syntax / method	Meaning	Example
list[i]	Accesses the item at index i (indexing starts at 0).	marks[0]
list[-1]	Accesses the last item.	marks[-1]
.append(x)	Adds an item to the end of a list.	marks.append(14)
.extend(seq)	Adds all items from an iterable.	a.extend(b)
.insert(i, x)	Inserts an item at index i.	a.insert(0, 10)
.remove(x)	Removes the first occurrence of x.	a.remove(10)
.pop()	Removes and returns an item (last item by default).	a.pop()
.sort()	Sorts a list in place.	a.sort()
sorted(seq)	Returns a new sorted list.	sorted(marks)
del	Deletes a variable or an item.	del marks[0]
{key: value}	Creates a dictionary.	{'name': 'Lina'}
.keys() / .values() / .items()	Accesses dictionary keys / values / key-value pairs.	student.items()
.get(key)	Gets a value safely if a key is missing.	student.get('age')
.update()	Adds or updates dictionary entries.	d.update({'age': 17})
.add(x) / .discard(x)	Adds / removes an item from a set.	colors.add('green')

9. Comprehensions and useful expressions

Syntax	Meaning	Example
[expression for x in sequence]	Builds a list from an iteration.	[x*x for x in range(5)]
{key: value for ...}	Builds a dictionary using a comprehension.	{x: x*x for x in range(4)}
{x for x in sequence}	Builds a set using a comprehension.	{x for x in [1,1,2]}
value_if_true if condition else value_if_false	Conditional expression on one line.	'even' if n%2==0 else 'odd'
any(sequence)	Returns True if at least one item is true.	any([False, True])
all(sequence)	Returns True if all items are true.	all([True, True])
map(function, sequence)	Applies a function to each item.	list(map(str, [1,2]))

filter(function, sequence)	Keeps items accepted by a function.	list(filter(bool, values))

10. Error and exception handling

Syntax	Meaning	Example
try	Starts a block that may raise an exception.	try:
except	Handles a specific or general exception.	except ValueError:
else	Runs if no exception occurred in the try block.	else:
finally	Runs whether an exception occurred or not.	finally:
raise	Raises an exception deliberately.	raise ValueError('Invalid value')
with	Manages a resource automatically, such as a file.	with open(...) as f:

11. Files and folders

Syntax / method	Meaning	Example
open(path, 'r')	Opens a file for reading.	open('text.txt', 'r')
'w'	Write mode: creates a file or overwrites its content.	open('text.txt', 'w')
'a'	Append mode: adds content to the end of a file.	open('text.txt', 'a')
'x'	Creates a new file; raises an error if it already exists.	open('new.txt', 'x')
.read()	Reads the whole file content.	f.read()
.readline()	Reads one line.	f.readline()
.write()	Writes text to a file.	f.write('Hello')
.close()	Closes a file (often handled automatically by with).	f.close()
pathlib.Path	Modern tools for working with file-system paths.	from pathlib import Path

12. Classes and object-oriented programming

Syntax	Meaning	Example
class	Defines a class (a blueprint for objects).	class Car:
__init__	Initializes a new object.	def __init__(self, brand):
self	Reference to the current instance.	self.brand = brand
object = Class()	Creates an instance of a class.	car = Car('Renault')
Inheritance	Allows a class to reuse features of another class.	class SportsCar(Car):
super()	Accesses a parent-class method.	super().__init__(brand)
@property	Makes a method accessible like an attribute.	@property
@staticmethod	Defines a method without automatic self or cls.	@staticmethod
@classmethod	Defines a method receiving the class as its first argument.	@classmethod

13. Modules, libraries and environments

Module / command	Meaning	Example
math	Mathematical functions: square roots, pi, trigonometry, etc.	math.sqrt(25)
random	Random numbers and selections.	random.randint(1, 6)
time	Time values and delays.	time.sleep(1)
datetime	Dates and times.	datetime.date.today()

statistics	Basic statistical calculations.	statistics.mean(marks)
os	Operating-system interaction.	os.getcwd()
pathlib	File-system path handling.	Path('course').exists()
json	Reads and writes JSON data.	json.dumps(data)
csv	Reads and writes CSV files.	csv.reader(f)
turtle	Drawing and beginner graphics.	turtle.forward(100)
tkinter	Creates graphical user interfaces.	import tkinter as tk
matplotlib	Creates charts and plots.	import matplotlib.pyplot as plt
numpy	Numerical computing with arrays.	import numpy as np
pandas	Data analysis and table manipulation.	import pandas as pd
pip install package	Installs a Python package from the terminal.	pip install matplotlib
python file.py	Runs a Python file from the terminal.	python program.py

14. Common built-in functions

Function	Meaning	Example
print()	Displays one or more values.	print('Hello')
input()	Reads keyboard input as text.	first_name = input()
len()	Returns the length of an object.	len([2,4,6])
range()	Produces a sequence of integers.	range(5)
sum()	Returns the sum of values.	sum([1,2,3])
min() / max()	Returns the minimum / maximum value.	max(marks)
abs()	Returns the absolute value.	abs(-8)
round()	Rounds a number.	round(3.14159, 2)
sorted()	Returns a sorted list.	sorted([3,1,2])
reversed()	Returns an iterator in reverse order.	list(reversed([1,2,3]))
enumerate()	Adds a counter to an iterable.	enumerate(names)
zip()	Combines items from multiple iterables.	zip(names, marks)
help()	Displays help about an object.	help(str)
dir()	Lists available attributes.	dir(list)
id()	Returns an object's identity during its lifetime.	id(object)

15. Python reserved keywords

These words have a special meaning in Python and cannot be used as variable names.

Keyword	Meaning
False	Boolean false value
True	Boolean true value
None	No value
and	Logical AND
or	Logical OR
not	Logical NOT
if	Conditional statement
elif	Else-if condition
else	Alternative branch
for	Loop over items
while	Loop while a condition is true
break	Exit a loop
continue	Skip to the next loop iteration
pass	Do nothing
def	Define a function
return	Return a value
class	Define a class
try	Start exception handling
except	Catch an exception
finally	Always run this block
raise	Raise an exception
import	Import a module
from	Import from a module

as	Create an alias
with	Use a context manager
lambda	Anonymous function
yield	Produce a generator value
global	Declare a global variable
nonlocal	Refer to an enclosing-scope variable
assert	Check a condition
del	Delete a name or item
in	Membership test
is	Object identity test
async	Declare asynchronous code
await	Wait for an asynchronous result
match	Match a value against patterns
case	Define a pattern case

16. Complete commented program example

```
# Program: calculate a student's average

name = input("Student name: ")
marks = []

for i in range(3):
    mark = float(input(f"Mark {i + 1}: "))
    marks.append(mark)

average = sum(marks) / len(marks)

print(f"Student: {name}")
print(f"Average: {average:.2f} / 20")

if average >= 10:
    print("Pass")
else:
    print("Needs improvement")
```

17. Good programming practices

- Choose meaningful names: class average is clearer than x.
- Use four spaces for indentation; do not mix tabs and spaces.
- Write one instruction per line to make code easier to read.
- Add useful comments, but do not comment on every obvious line.
- Convert user input when necessary: input() always returns a string.
- Test programs with different values, including edge cases.
- Read error messages carefully: they often show the file, line and error type.
- Use the official documentation: https://docs.python.org/3/.

18. Mini glossary

Term	Definition
Statement	An instruction executed by Python.
Variable	A named place used to store a value.
Value	The data stored or processed.
Data type	The category of a value, such as int or str.
String	A sequence of text characters.

Integer	A whole number.
Floating-point number	A number that can contain a decimal part.
Condition	An expression evaluated as True or False.
Loop	A structure that repeats instructions.
Function	A reusable block of code.
Argument	A value passed to a function call.
Parameter	A variable in a function definition.
List	An ordered, mutable collection.
Dictionary	A collection of key-value pairs.
Index	The position of an item in a sequence.
Error / exception	An event that interrupts normal program execution.
File	A named collection of data stored on a device.
Library	A collection of reusable code.
Module	A Python file that can be imported.
Debugging	The process of finding and fixing errors.