

PYTHON 3 クイックリファレンス

基礎から pandas、requests、csv、json まで

基礎

変数

```
name = "Alice" # str
age = 20 # int
gpa = 3.85 # float
active = True # bool
```

データ型

```
str 文字列: `hello`
int 整数: `42`
float 小数: `3.14`
bool True / False
list 順序付き可変: `[1, 2, 3]`
tuple 順序付き不変: `(1, 2)`
dict キーと値: `{"a": 1}`
set ユニーク要素: `{1, 2, 3}`
```

算術演算

```
+ - * 加算、減算、乗算
/ 除算 (浮動小数点): `7/2` → 3.5
// 切り捨て除算: `7//2` → 3
% 剰余: `7%2` → 1
** べき乗: `2**3` → 8
```

型変換

```
int("42") # 42
float("3.14") # 3.14
str(100) # "100"
list("abc") # ['a', 'b', 'c']
```

ユーザー入力

```
name = input("Your name? ")
age = int(input("Age? "))
```

文字列

文字列の作成

```
s1 = 'single quotes'
s2 = "double quotes"
s3 = """triple quotes
for multiline"""
```

f文字列 (Python 3.6+)

```
name = "Alice"
f"Hello, {name}!" # Hello, Alice!
f"{2 + 3}" # 5
f"{3.14159:.2f}" # 3.14
f"{1000:,}" # 1,000
```

文字列スライス

```
s = "Python"
# index: 0 1 2 3 4 5
s[0] # 'P'
s[-1] # 'n'
s[2:5] # 'thy'
s[:2] # 'Py'
s[2:] # 'thon'
s[::-1] # 'nohtyP' (reverse)
```

文字列メソッド

```
len(s) 文字列の長さ
s.upper() 大文字に変換
s.lower() 小文字に変換
s.strip() 前後の空白を削除
s.split(" ") リストに分割
", ".join(lst) リストを文字列に結合
s.replace(a, b) aをbに置換
s.find("x") 最初のマッチのインデックス (なければ-1)
s.startswith(x) 前方一致チェック → bool
s.endswith(x) 後方一致チェック → bool
s.count(x) 出現回数
"x" in s 含まれるかチェック → bool
```

リスト

作成 & アクセス

```
fruits = ["apple", "banana", "cherry"]
fruits[0] # "apple"
fruits[-1] # "cherry"
fruits[1:3] # ["banana", "cherry"]
```

リスト内包表記

```
squares = [x**2 for x in range(5)]
# [0, 1, 4, 9, 16]
evens = [x for x in range(10) if x%2==0]
# [0, 2, 4, 6, 8]
```

リストメソッド

```
lst.append(x) 末尾に追加
lst.extend(lst2) lst2の要素を全て追加
lst.insert(i, x) インデックスiに挿入
lst.pop() 末尾を削除して返す
lst.pop(i) i番目を削除して返す
lst.remove(x) 最初のxを削除
del lst[i] インデックスで削除
lst.sort() インプレースでソート
sorted(lst) ソート済みのコピーを返す
lst.reverse() インプレースで逆順
len(lst) 要素数
x in lst メンバシップチェック
lst.index(x) xの最初のインデックス
lst.count(x) xの出現回数
```

タプル & セット

タプル (不変)

```
point = (3, 4)
x, y = point # unpacking
point[0] # 3 (read-only)
```

セット (ユニーク要素)

```
s = {1, 2, 3}
s.add(4); s.remove(1)
a & b # intersection
a | b # union
a - b # difference
```

辞書

作成 & アクセス

```
student = {"name": "Alice", "age": 20}
student["name"] # "Alice"
student.get("gpa", 0) # 0 (default)
student["gpa"] = 3.85 # add/update
```

辞書内包表記

```
sq = {x: x**2 for x in range(5)}
# {0:0, 1:1, 2:4, 3:9, 4:16}
```

イテレーション

```
for k, v in student.items():
    print(f"{k}: {v}")
```

辞書メソッド

```
d.keys() 全キー
d.values() 全値
d.items() 全(キー、値)ペア
d.get(k, default) デフォルト付きで取得
d.update(d2) d2をdにマージ
d.pop(k) キーを削除して値を返す
del d[k] キーを削除
"key" in d キーの存在チェック → bool
len(d) エントリ数
```

制御フロー

if / elif / else

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
```

三項演算子

```
status = "pass" if score >= 60 else "fail"
```

ループ

for ループ

```
for fruit in ["apple", "banana"]:
    print(fruit)
```

range()

```
range(5) # 0, 1, 2, 3, 4
range(2, 5) # 2, 3, 4
range(0, 10, 2) # 0, 2, 4, 6, 8
```

while ループ

```
while count < 10:
    count += 1
```

enumerate() & zip()

```
for i, val in enumerate(["a", "b"]):
    print(i, val) # 0 a, 1 b

for a, b in zip([1, 2], ["x", "y"]):
    print(a, b) # 1 x, 2 y
```

break & continue

```
for x in range(10):
    if x == 5: break # stop loop
    if x % 2 == 0: continue # skip
```

関数

定義 & 呼び出し

```
def greet(name, greeting="Hi"):
    return f"{greeting}, {name}!"
```

```
greet("Alice") # "Hi, Alice!"
greet("Bob", "Hello") # "Hello, Bob!"
```

複数の戻り値

```
def min_max(lst):
    return min(lst), max(lst)
lo, hi = min_max([3, 1, 4, 1, 5])
```

*args & **kwargs

```
def total(*args): # args is a tuple
    return sum(args)
total(1, 2, 3) # 6
```

```
def info(**kwargs): # kwargs is a dict
    print(kwargs)
```

ラムダ関数

```
square = lambda x: x**2
square(5) # 25
sorted(lst, key=lambda x: x["age"])
```

クラス

```
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        return f"{self.name} says Woof!"

dog = Dog("Rex", "Lab")
dog.bark() # "Rex says Woof!"
```

継承

```
class Puppy(Dog):
    def __init__(self, name, breed, toy):
        super().__init__(name, breed)
        self.toy = toy
```

エラーハンドリング

```
try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
finally:
    print("Always runs")
```

ファイル I/O

ファイルの読み込み

```
with open("data.txt") as f:
    content = f.read() # full text
```

```
with open("data.txt") as f:
    for line in f: # line by line
        print(line.strip())
```

ファイルへの書き込み

```
with open("out.txt", "w") as f:
    f.write("Hello\n")
```

"r" = 読み込み "w" = 書き込み (上書き) "a" = 追記

CSV

```
import csv

with open("data.csv") as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row["name"])
```

```
with open("out.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["name", "age"])
```

JSON

```
import json

data = json.loads('{ "name": "Alice" }') # parse
text = json.dumps(data) # serialize

with open("data.json") as f:
    data = json.load(f) # read file
with open("out.json", "w") as f:
    json.dump(data, f, indent=2) # write file
```

HTTP リクエスト

```
import requests

# GET
r = requests.get("https://api.example.com/data")
r.status_code # 200
data = r.json() # parse JSON

# POST
r = requests.post(url, json={"key": "val"})
```

pandas 基礎

```
import pandas as pd
df = pd.read_csv("data.csv")
df.head() # first 5 rows
df.shape # (rows, cols)
df["name"] # single column
df[df["age"] > 20] # filter rows
```

便利な組み込み関数

```
print() コンソールへ出力
len() 長さ / 要素数
type() オブジェクトの型
range() 数値のシーケンス
enumerate() インデックスと値のペア
zip() イテラブルの要素をペアに
sorted() 定義 & 呼び出し
sum() min() max() 集計関数
```

モジュール

```
import math
from math import sqrt, pi
import pandas as pd # alias
```