FREE CHEAT SHEET

Python Quick Reference, for Singapore CS Students

Two pages. The patterns most NUS, NTU, and polytechnic Python assignments actually need, plus the bugs that quietly cost you marks.

COMMON PATTERNS

Loops over indices vs values

```
# by value
for name in students:
    print(name)

# by index + value
for i, name in enumerate(students):
    print(i, name)
```

List comprehensions

```
squares = [x*x for x in nums]
evens   = [x for x in nums if x % 2 == 0]
matrix  = [[r*c for c in range(3)]
           for r in range(3)]
```

Slicing

```
a[2:5]    # indices 2,3,4
a[:3]     # first three
a[-2:]    # last two
a[::-1]   # reversed
a[::2]    # every second
```

Dict patterns

```
counts.get(key, 0)         # default
counts[key] = counts.get(key, 0) + 1

from collections import Counter
Counter(words).most_common(5)
```

FIVE BUGS THAT COST MARKS

1. Mutating a list while iterating

Removing or inserting while looping shifts indices and Python skips elements.

```
# wrong
for n in nums:
    if n % 2 == 0: nums.remove(n)

# right
nums = [n for n in nums if n % 2]
```

2. Mutable default arguments

Default `[]` is created once when the function is defined, not on every call.

```python
# wrong
def add(name, items=[]):
    items.append(name)
    return items

# right
def add(name, items=None):
    if items is None: items = []
    items.append(name)
    return items
```

3. `==` vs `is`

`==` compares values, `is` compares identity. Use `is` only for `None`, `True`, `False`.

```python
a = [1, 2]; b = [1, 2]
a == b   # True
a is b   # False

if x is None: ...   # correct
```

4. Strings are immutable

In-place character swaps fail. Build a new string.

```python
# wrong
s[0] = 'X'   # TypeError

# right
s = 'X' + s[1:]
s = s[::-1]   # reverse
```

5. Implicit globals in functions

Assigning to a name inside a function makes it local. Either declare `global` or pass values in and out.

```python
# wrong
count = 0
def inc():
    count = count + 1   # UnboundLocalError

# better
def inc(count):
    return count + 1
```

DEBUGGING IN 4 STEPS

1. **Read the error**, the line number and message often tell you exactly what's wrong.
2. **State your hypothesis** in one sentence before changing any code.
3. **Reduce** until you have the smallest program that still misbehaves. Bugs reveal themselves.

4. **Use** `breakpoint()` instead of print statements. `n` next line, `s` step in, `c` continue.

MODULES WHERE THIS SAVES YOU THE MOST MARKS

NUS CS1010 (intro), CZ1003 NTU (Python intro), SC1003 NTU, IS200 SMU, INF1002 SIT, polytechnic Programming Fundamentals, DSA1101 NUS (R but same patterns), data analytics modules across all polys.