Python Cheat Sheet – Part 1

---

1. Arithmetic Operators

Operator	Description	Example
`**`	Exponentiation	`2 ** 3 → 8`
`*`	Multiplication	`3 * 4 → 12`
`/`	Division (always float)	`7 / 2 → 3.5`
`//`	Floor division (integer)	`7 // 2 → 3`
`%`	Modulo (remainder)	`7 % 2 → 1`
`+`	Addition	`5 + 3 → 8`
`-`	Subtraction	`5 - 3 → 2`
`()`	Parentheses (order of ops)	`(2 + 3) * 4 → 20`

---

2. Data Types

Type	Example	Notes
`int`	5	Whole numbers
`float`	3.14	Decimal numbers
`str`	`"hello"`	Immutable sequence of characters
`bool`	`True` / `False`	Logical values
`tuple`	`(1, 2, 3)`	Immutable ordered collection
`list`	`[1, 2, 3]`	Mutable ordered collection
`dict`	`{'a':1, 'b':2}`	Key-value pairs, mutable

---

3. Loops

Loop examples combined:

```
# While loop
x = 0
while x < 3:
    print(x)
    x += 1

# For loop
for i in range(3):
    print(i)
```

Loop control statements

Statement	Purpose
`pass`	Placeholder, does nothing
`continue`	Skip to next iteration
`break`	Exit loop immediately

---

4. Branching (Conditional Statements)

If/elif/else combined example:

```
x = 10
if x > 5:
    print("x > 5")
elif x == 5:
    print("x = 5")
else:
    print("x < 5")
```

Logical & Comparison Operators

Type	Operators
Comparison	`==, !=, <, >, <=, >=`
Membership	`in, not in`
Logical	`and, or, not`

---

5. Exception Handling (`try` blocks)

Combined example:

```
try:
    num = int(input("Enter a number: "))
except ValueError:
    print("That's not a number!")
else:
    print(f"You entered {num}")
finally:
    print("Execution complete")
```

- `try`: code that might raise an exception
- `except`: handle exceptions (e.g., `ValueError`, `Exception`)
- `else`: runs if no exception occurred
- `finally`: always runs

---

6. Generic Functions

All function examples combined:

```
# Defining a function
def my_function(param1, param2):
    """
    Optional docstring: describes what the function does.
    """
    result = param1 + param2
    return result

# Function types
def greet():
    print("Hi")  # No parameters, no return

def print_sum(a,b):
    print(a+b)   # With parameters, no return

def add(a,b):
    return a+b   # With parameters and return

# Parameters & arguments
def f(a, b):
    return a - b
f(5,2)  # Positional arguments

f(b=2, a=5)  # Keyword arguments

def greet(name="Guest"):
    print(f"Hello {name}")  # Default value
greet()

# Mutable vs immutable behavior
def change_num(x):
    x += 5
n = 10
change_num(n)
print(n)  # still 10 (immutable)

def add_item(lst):
    lst.append(4)
numbers = [1,2,3]
add_item(numbers)
print(numbers)  # [1,2,3,4] (mutable)
```

Tips:

- Use **docstrings** to describe what a function does: `"""This function does X."""`
- Prefer **immutable parameters** for safety if you don't want side effects.
- Return values for computations; use `print` for displaying results.