

# ■ Python Basics — Detailed Notes

## 1. Print & Comments

`print("Hello, World!")` # prints text to the console

Example:

`print("Hi")` -> Output: Hi

Comments:

# single-line comment

"""triple single quotes as a multi-line comment or docstring"""

## 2. Variables & Data Types

Variables hold values; types are inferred automatically.

Examples:

`x = 10` # int

`pi = 3.14` # float

`name = "Bashu"` # str

`flag = True` # bool

Check type:

`type(x)` ->

## 3. Input & Conversions

Input from user is always a string.

`name = input("Name: ")` # returns text

`age = int(input("Age: "))` # convert to integer

Be careful: `int("abc")` raises `ValueError`.

Use `try/except` to handle invalid input.

## 4. Operators (short)

Arithmetic: `+` `-` `*` `/` `%` `//` `**` (floor division, power)

Comparison: `==` `!=` `>` `<` `>=` `<=`

Logical: `and` `or` `not`

Example:

`5 // 2` -> 2

`2 ** 3` -> 8

## 5. Strings — slicing & common methods

`s = "Python"`

`s[0]` -> 'P'

`s[-1]` -> 'n'

`s[0:4]` -> 'Pyth'

`len(s)` -> 6

Common methods:

`s.upper()` -> "PYTHON"

`s.lower()` -> "python"

`s.replace("Py", "My")` -> "Mython"

```
s.split("t") -> ['Py', 'hon']
```

f-strings:

```
name = "Asha"
```

```
f"Hello {name}" -> "Hello Asha"
```

## 6. Lists — mutable sequences

```
fruits = ["apple", "banana", "mango"]
```

```
fruits.append("orange") # add
```

```
fruits.remove("banana") # remove by value
```

```
fruits.pop() # remove last and return it
```

```
fruits[0] -> "apple"
```

```
fruits[1:3] -> slice of list
```

Useful: `sorted(fruits)`, `fruits.sort()`, `reversed(list(fruits))`

## 7. Tuples & Sets

Tuple: immutable ordered collection

```
point = (3, 4)
```

```
point[0] -> 3
```

Set: unordered collection of unique values

```
s = {1, 2, 2, 3} -> {1, 2, 3}
```

Methods: `add()`, `remove()`, `union()`, `intersection()`

## 8. Dictionaries

Key-value mapping:

```
student = {"name": "Ashwin", "age": 20}
```

```
student["name"] -> "Ashwin"
```

```
student["grade"] = "A" # insert/update
```

Iterate:

```
for k in student: # keys
```

```
for k, v in student.items(): # key, value pairs
```

## 9. Conditionals

```
if / elif / else
```

```
age = 18
```

```
if age >= 18:
```

```
print("Adult")
```

```
elif age > 12:
```

```
print("Teenager")
```

```
else:
```

```
print("Child")
```

## 10. Loops & loop tools

```
for i in range(1, 6): # 1..5
```

```
print(i)
```

while loop:

```
i = 1
while i <= 5:
    print(i)
    i += 1
```

Useful patterns:  
for idx, val in enumerate(my\_list): # get index + value  
...

Loop control:  
break # exit loop  
continue # skip to next iteration

## 11. Functions

```
def greet(name):
    """Return greeting for name (docstring example)."""
    return "Hello " + name
```

```
# default args
def add(a, b=5):
    return a + b
```

```
# variable args
def f(*args, **kwargs):
    pass
```

## 12. Comprehensions & Lambdas

List comprehension:  
squares = [x\*x for x in range(5)] # [0,1,4,9,16]

Dictionary comprehension:  
d = {x: x\*x for x in range(5)}

Lambda:  
double = lambda x: x\*2  
list(map(lambda x: x\*2, range(5))) -> [0,2,4,6,8]

## 13. Modules & common stdlib

Importing:  
import math  
math.sqrt(16) -> 4.0

from random import choice  
choice([1,2,3]) -> random element

Other useful modules: os, sys, json, datetime

## 14. File I/O (basic)

Write text:  
with open("out.txt", "w", encoding="utf-8") as f:  
 f.write("Hello\n")

Read:  
with open("out.txt", "r", encoding="utf-8") as f:  
data = f.read()

## 15. Exceptions (try/except)

```
try:  
x = int(input("Enter number: "))  
except ValueError:  
print("Please enter a number")  
finally:  
print("This always runs")
```

## 16. Useful built-ins & tips

len(), sum(), min(), max(), sorted()  
range(start, stop, step)  
help(list) # shows documentation in REPL  
Use virtual environments for projects:  
python -m venv venv  
source venv/bin/activate (or venv\Scripts\activate on Windows)