pythonquickrevision

January 26, 2026

[52]:
```python
#  F-strings (formatted printing)
name = "Tarun"
age = 21
height = 5.9821

# simple f-string
print(f"My name is {name} and I am {age} years old.")

# round decimal inside f-string
print(f"My height is {height:.2f} feet.")  # 5.68
print(f"My height (3 decimals) is {height:.3f}")  # 5.679

# add commas
num = 1234567.8912
print(f"Formatted number: {num:,.2f}")  # 1,234,567.89
```

```
My name is Tarun and I am 21 years old.
My height is 5.98 feet.
My height (3 decimals) is 5.982
Formatted number: 1,234,567.89
```

[53]:
```python
#  Rounding numbers
import math
x = 3.14159
print(round(x))  # 3
print(round(x, 2))  # 3.14
print(round(x, 3))  # 3.142
print(math.ceil(x))  # 4
print(math.floor(x))  # 3

import random
print(random.randint(1, 10))  # random int between 1 and 10 inclusive
print(random.randrange(0, 21, 2))  # random even int between 0 and 20 inclusive
```

```
3
3.14
3.142
4
3
```

```
10
20
```

[54]:
```python
# map() function
nums = [1, 2, 3, 4, 5]

# double each element
doubled = list(map(lambda x: x * 2, nums))
print("Doubled:", doubled)

# convert list of strings to integers
str_nums = ["10", "20", "30"]
int_nums = list(map(int, str_nums))
print("Integers:", int_nums)
```

```
Doubled: [2, 4, 6, 8, 10]
Integers: [10, 20, 30]
```

[55]:
```python
#  filter() function
# keep only even numbers
evens = list(filter(lambda x: x%2==0, nums))
print("Even numbers:", evens)
```

```
Even numbers: [2, 4]
```

0.0.1 Lists

[56]:
```python
# Lists
# Empty list
lst = []
# List with values
lst = [1, 2, 3, 4]
l2=['a', 'b', 'c']
l3=[1, 'a', 3.5, True]  # mixed types
# Using list() from iterable
lst2 = list(range(5))  # [0,1,2,3,4]

# access/update
print(lst[0])  # 1
lst[2] = 10  # [1,2,10,4]

# add/remove
lst.append(5)  # add at end
lst.insert(2, 99)  # insert at index
lst.pop()  # remove last
lst.pop(1)  # remove index 1
lst.remove(10)  # remove by value
lst.extend([6,7,8])  # extend list
lst.reverse()  # reverse list
```

```
lst.sort()  # sort list


# common methods
len(lst)  # length
sum(lst)  # sum of elements
min(lst), max(lst)  # min/max

#  List comprehensions (alternative to map/filter)
squares = [x * x for x in nums]
even_squares = [x * x for x in nums if x % 2 == 0]
print("Squares:", squares)
print("Even squares:", even_squares)

# check empty list:
if not lst:
    print("List is empty")

lst2=lst[:] # copy list or lst.copy()
```

```
1
Squares: [1, 4, 9, 16, 25]
Even squares: [4, 16]
```

[57]:
```
# sorting
lst.sort()  # ascending
lst.sort(reverse=True)  # descending
lst.sort(key=lambda x: -x)  # custom key (negative for descending)
lst.sort(key=lambda x: x%3)  # custom key (mod 3)

# Sort by key, example: list of tuples by second element
tuples = [(1, 3), (2, 2), (3, 1)]
tuples.sort(key=lambda x: x[1])  # [(3,1),(2,2),(1,3)]
```

0.0.2 Dictionary

[58]:
```
# Initialize
d = {}  # empty
d = {"a": 1, "b": 2}

# from list of tuples
d2 = dict([("x", 10), ("y", 20)])

# access/update
print(d["a"])  # 1
d["c"] = 3  # add new
d["a"] = 10  # update existing
```

```
d.get("b")  # 2
d.get("z", 0)  # 0 (default if not found)

# useful methods
d.keys()  # dict_keys(['a','b','c'])
d.values()  # dict_values([10,2,3])
d.items()  # dict_items([('a',10),('b',2),('c',3)])
```

```
1
```

[58]:
```
dict_items([('a', 10), ('b', 2), ('c', 3)])
```

[59]:
```
# Iteration
for key in d:
    print(key, d[key])

for key, val in d.items():
    print(key, val)
```

```
a 10
b 2
c 3
a 10
b 2
c 3
```

[60]:
```
# counting
from collections import Counter

words = ["a", "b", "a", "c", "b", "a"]
count = Counter(words)
print(count)  # Counter({'a':3, 'b':2,'c':1})
```

```
Counter({'a': 3, 'b': 2, 'c': 1})
```

[61]:
```
# Max Frequency Word Example
max_freq = max(count.values())
candidates = [w for w, f in count.items() if f == max_freq]
longest_word = max(candidates, key=len)
```

0.0.3 Strings

[62]:
```
# initialize/access
s = "Hello World"
s[0]  # 'H'
s[-1]  # 'd'
len(s)  # 11
s[1:9]
s[1:9:2]  # 'el ol'
```

```
s[-5:] # 'World'
s[:: -1]  # 'dlroW olleH' (reversed)
```

[62]:
```
'dlroW olleH'
```

[63]:
```
# Common methods
s.lower()
s.upper()
s.strip()  # remove spaces
s.split()  # ['Hello','World']
s.split(",")  # split by comma
",".join(["a", "b"])  # 'a,b'
s.replace("l", "L")  # 'HeLLo WorLd'

# check/search
s.startswith("He")
s.endswith("ld")
"lo" in s
s.count("l")

# all method
all(c.isalpha() for c in s)  # checks if all chars are alphabetic
s.isdigit()  # checks if all chars are digits
s.isalpha()  # checks if all chars are alphabetic
s.isalnum()  # checks if all chars are alphanumeric
s.isspace()  # checks if all chars are whitespace
```

[63]:
```
False
```

0.0.4 Tuple

[64]:
```
# Initialize
t = (10, 20, 30)
t_single = (50,)  # single element tuple, comma needed
print(t)
print(t_single)

# access/operations
t[0]  # 10
t += (40,)  # create new tuple
len(t)
```

```
(10, 20, 30)
(50,)
```

[64]:
```
4
```

0.0.5 Sets

[65]:
```
# Initialise
s = set()
s = {1, 2, 3}

# operations
s.add(4)
s.remove(2)
len(s)
1 in s
```

[65]:
```
True
```

[66]:
```
# useful set operations
a = {1, 2, 3}
b = {2, 3, 4}
a & b  # intersection {2,3}
a | b  # union {1,2,3,4}
a - b  # difference {1}
```

[66]:
```
{1}
```

Common Patterns / Quick Snippets

[67]:
```
## Counting letters in string
from collections import Counter

s = "abracadabra"
count = Counter(s)
```

[68]:
```
## Reverse words / characters
word = "hello"
rev = word[::-1]  # 'olleh'

sentence = "hi there"
rev_sentence = " ".join(w[::-1] for w in sentence.split())
```

[69]:
```
# Stable partition (vowels / consonants)
s = "program"
vowels = "aeiouAEIOU"
consonants = [c for c in s if c not in vowels]
v = [c for c in s if c in vowels]
"".join(consonants + v)
```

[69]:
```
'prgrmoa'
```

[70]:
```
# Max element / tie-breaker example
words = ["hi", "hello", "hey", "abc"]
```

```
max_len = max(len(w) for w in words)
longest = [w for w in words if len(w) == max_len]
print(longest)
```

```
['hello']
```

0.0.6 Regex

[71]:
```
import re

text = "Hello123, my email is example@test.com and phone is 9876543210."

# ----------------------
# 1 Basic anchors
# ^ → start of string
# $ → end of string

print(re.findall(r"^Hello", text))  # ['Hello']  → matches "Hello" only at start
print(re.findall(r"com$", text))  # [] → 'com' is not at end
print(re.findall(r"10\.$", text))  # ['10.'] → matches number at very end

# ----------------------
# 2 Character classes
# [abc] → any one of a, b, or c
# [a-z] → any lowercase letter
# [A-Z] → any uppercase letter
# [0-9] → any digit
# [^a-z] → any character NOT a lowercase letter

print(re.findall(r"[aeiou]", text))  # ['e', 'o', 'a', 'i', 'e', 'a', 'e', 'e',␣
↪'o']
print(
    re.findall(r"[0-9]", text)
)  # ['1','2','3','9','8','7','6','5','4','3','2','1','0']
print(re.findall(r"[^0-9]", text))  # all non-digit characters

# ----------------------
# 3 Quantifiers
# *  → 0 or more
# +  → 1 or more
# ?  → 0 or 1
# {n} → exactly n times
# {n,} → n or more times
# {m,n} → between m and n times

print(re.findall(r"\d+", text))  # ['123', '9876543210'] → 1+ digits
```

```
print(re.findall(r"\d{3}", text))  # ['123','987','654','321'] → exactly 3␣
 ↪digits
print(
    re.findall(r"\w+", text)
)  #␣
 ↪['Hello123','my','email','is','example','test','com','and','phone','is','9876543210']

# ----------------------
# 4  Special sequences
# \d → digit [0-9]
# \D → non-digit
# \w → word character [a-zA-Z0-9_]
# \W → non-word character
# \s → whitespace
# \S → non-whitespace

print(re.findall(r"\s", text))  # [' ', ' ', ' ', ' ', ' ', ' ']
print(
    re.findall(r"\S+", text)
)  # ['Hello123,','my','email','is','example@test.
 ↪com','and','phone','is','9876543210.']

# ----------------------
# 5  Common real-world patterns

# Email
emails = re.findall(r"\b[\w._%+-]+@[\w.-]+\.\w{2,}\b", text)
print("Emails:", emails)  # ['example@test.com']

# Phone (10 digits)
phones = re.findall(r"\b\d{10}\b", text)
print("Phones:", phones)  # ['9876543210']

# Words starting with uppercase
capital_words = re.findall(r"\b[A-Z][a-z]*\b", text)
print("Capitalized words:", capital_words)  # ['Hello']

# Remove punctuation (example)
cleaned = re.sub(r"[^\w\s]", "", text)
print("Cleaned:", cleaned)

# ----------------------
# 6  Optional / OR
# ? → optional
# | → OR

# match 'cat' or 'dog'
```

```
print(re.findall(r"cat|dog", "I have cat and dog"))  # ['cat','dog']

# match 'color' or 'colour'
print(re.findall(r"colou?r", "color colour"))  # ['color','colour']
```

```
['Hello']
[]
['10.']
['e', 'o', 'e', 'a', 'i', 'i', 'e', 'a', 'e', 'e', 'o', 'a', 'o', 'e', 'i']
['1', '2', '3', '9', '8', '7', '6', '5', '4', '3', '2', '1', '0']
['H', 'e', 'l', 'l', 'o', ',', ' ', 'm', 'y', ' ', 'e', 'm', 'a', 'i', 'l', ' ',
'i', 's', ' ', 'e', 'x', 'a', 'm', 'p', 'l', 'e', '@', 't', 'e', 's', 't', '.',
'c', 'o', 'm', ' ', 'a', 'n', 'd', ' ', 'p', 'h', 'o', 'n', 'e', ' ', 'i', 's',
' ', '.']
['123', '9876543210']
['123', '987', '654', '321']
['Hello123', 'my', 'email', 'is', 'example', 'test', 'com', 'and', 'phone',
'is', '9876543210']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['Hello123,', 'my', 'email', 'is', 'example@test.com', 'and', 'phone', 'is',
'9876543210.']
Emails: ['example@test.com']
Phones: ['9876543210']
Capitalized words: []
Cleaned: Hello123 my email is exampletestcom and phone is 9876543210
['cat', 'dog']
['color', 'colour']
```

0.0.7 Regex Cheat Sheet – With search and sub

[72]:
```
import re

text = "Hello123, my email is example@test.com and phone is 9876543210."

# ----------------------
# 1  re.findall() → returns all matches as a list
emails = re.findall(r"\b[\w._%+-]+@[\w.-]+\.\w{2,}\b", text)
print("findall emails:", emails)  # ['example@test.com']

# ----------------------
# 2  re.search() → returns first match object (or None)
match = re.search(r"\d{10}", text)
if match:
    print("search phone:", match.group())  # '9876543210'

# ----------------------
# 3  re.sub() → substitute / replace pattern
```

```python
# Replace all digits with 'X'
masked = re.sub(r"\d", "X", text)
print("sub digits:", masked)

# Replace email with [EMAIL]
masked_email = re.sub(r"\b[\w._%+-]+@[\w.-]+\.\w{2,}\b", "[EMAIL]", text)
print("sub email:", masked_email)

# ----------------------
# 4  Anchors
print(re.findall(r"^Hello", text))  # ['Hello']  start
print(re.findall(r"10\.$", text))  # ['10.']  end

# ----------------------
# 5  Character classes
print(re.findall(r"[A-Z]", text))  # uppercase letters
print(re.findall(r"[0-9]", text))  # digits

# ----------------------
# 6  Quantifiers
print(re.findall(r"\d+", text))  # 1+ digits: ['123', '9876543210']
print(re.findall(r"\w+", text))  # word chars: ['Hello123','my','email','is',...
 ↪]

# ----------------------
# 7  Special sequences
print(re.findall(r"\s", text))  # whitespaces
print(re.findall(r"\S+", text))  # non-whitespaces

# ----------------------
# 8  Optional / OR
print(re.findall(r"colou?r", "color colour"))  # ['color','colour']
print(re.findall(r"cat|dog", "I have cat and dog"))  # ['cat','dog']

# ----------------------
# 9  Word boundaries
print(re.findall(r"\b\w{2}\b", "hi my is go at"))  # ['hi', 'my', 'is', 'go',␣
 ↪'at']

# ----------------------
#    Combining search & sub
# Find first email, replace all emails
match_email = re.search(r"\b[\w._%+-]+@[\w.-]+\.\w{2,}\b", text)
if match_email:
    print("First email found:", match_email.group())

new_text = re.sub(r"\b[\w._%+-]+@[\w.-]+\.\w{2,}\b", "[EMAIL]", text)
```

```
print("Text after replacement:", new_text)
```

```
findall emails: ['example@test.com']
search phone: 9876543210
sub digits: HelloXXX, my email is example@test.com and phone is XXXXXXXXXX.
sub email: Hello123, my email is [EMAIL] and phone is 9876543210.
['Hello']
['10.']
['H']
['1', '2', '3', '9', '8', '7', '6', '5', '4', '3', '2', '1', '0']
['123', '9876543210']
['Hello123', 'my', 'email', 'is', 'example', 'test', 'com', 'and', 'phone',
'is', '9876543210']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['Hello123,', 'my', 'email', 'is', 'example@test.com', 'and', 'phone', 'is',
'9876543210.']
['color', 'colour']
['cat', 'dog']
['hi', 'my', 'is', 'go', 'at']
First email found: example@test.com
Text after replacement: Hello123, my email is [EMAIL] and phone is 9876543210.
```

0.0.8 Regex Extraction in Python

[73]:
```
import re

text = "My name is Tarun, my email is example@test.com, phone: 9876543210, age:␣
↪21"

# ----------------------
# 1 Using groups to extract parts
# Extract username and domain from email
match = re.search(r"(\w+)@([\w.-]+)", text)
if match:
    print("Full match:", match.group(0))  # 'example@test.com'
    print("Username:", match.group(1))  # 'example'
    print("Domain:", match.group(2))  # 'test.com'

# ----------------------
# 2 Extract multiple matches with findall
# Returns list of tuples when groups used
emails = re.findall(r"(\w+)@([\w.-]+)", text)
print("Emails found (username, domain):", emails)  # [('example','test.com')]

# ----------------------
# 3 Named groups (easier to read)
match = re.search(r"(?P<user>\w+)@(?P<domain>[\w.-]+)", text)
```

```python
if match:
    print("Username:", match.group("user"))
    print("Domain:", match.group("domain"))

# ----------------------
# 4 Extract numbers
numbers = re.findall(r"\d+", text)
print("Numbers found:", numbers)  # ['9876543210', '23']

# ----------------------
# 5 Extract phone specifically
phone_match = re.search(r"phone: (\d{10})", text)
if phone_match:
    print("Phone number extracted:", phone_match.group(1))

# ----------------------
# 6 Extract all key-value patterns (like age: 23)
kv_pairs = re.findall(r"(\w+): (\d+)", text)
print("Key-value pairs:", kv_pairs)  # [('phone','9876543210'),('age','23')]

# ----------------------
# 7 Using sub to extract & replace
# Replace all numbers with [NUM]
masked_text = re.sub(r"\d+", "[NUM]", text)
print("Text after masking numbers:", masked_text)
```

```
Full match: example@test.com
Username: example
Domain: test.com
Emails found (username, domain): [('example', 'test.com')]
Username: example
Domain: test.com
Numbers found: ['9876543210', '21']
Phone number extracted: 9876543210
Key-value pairs: [('phone', '9876543210'), ('age', '21')]
Text after masking numbers: My name is Tarun, my email is example@test.com,
phone: [NUM], age: [NUM]
```

0.0.9 reference

https://chatgpt.com/share/69776e1c-b148-8002-af94-9e09e555c135