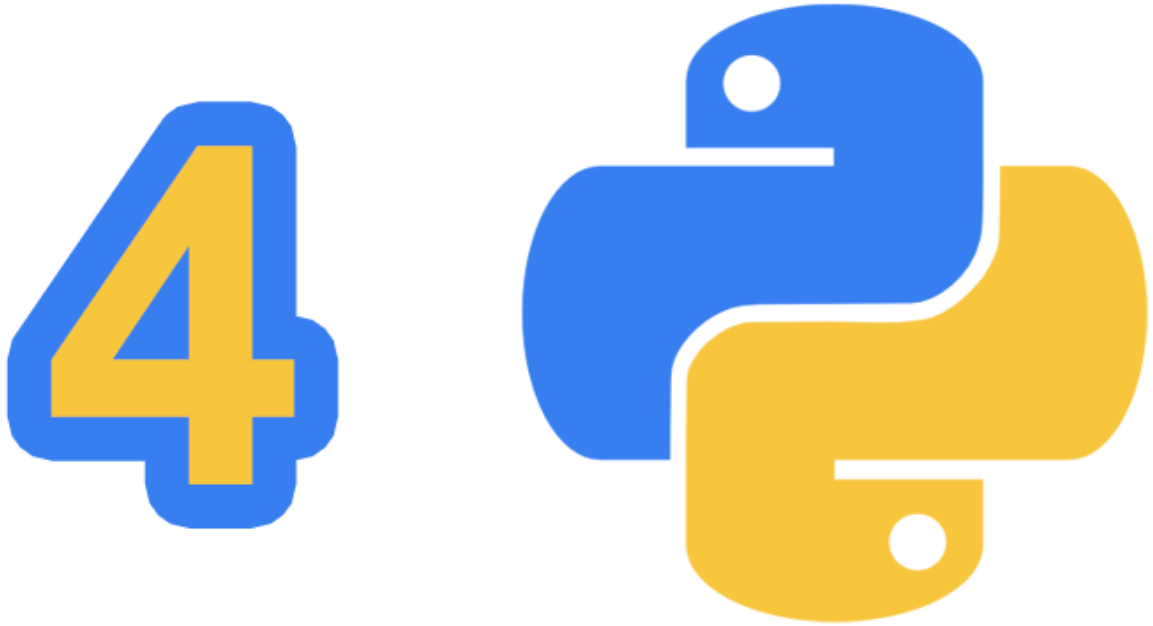


Python Quick Tips

Working with Strings

Python Quick Tips #4

Working with Strings

Strings

In part 1, we saw that **strings** are a data type

What are Strings

Output

String, 1, 5.4

Syntax

“String”, “1”, “5.4”

Or

‘String’, ‘1’, ‘5.4’

Convert to String

Function
str(object**)**

test.py

```
1  str1 = str(1)
2  str2 = str(2.5)
3  str3 = str(True)
4
5  print([str1, str2, str3])
```

Shell

```
>>> %Run test.py

  ['1', '2.5', 'True']
```

‘ vs. “

Be mindful of characters
‘ and “ in strings

test.py

```
1  print("stri'ng")
```

Shell

```
>>> %Run test.py

  stri'ng

>>> |
```

test.py

```
1  print('stri'ng')
```

Shell

```
>>> %Run test.py
  Traceback (most recent call last):
    File "C:\Users\Gavin\Desktop\test
  .py", line 1
      print('stri'ng')
                   ^
  SyntaxError: invalid syntax

>>> |
```

Escape Sequences

Syntax

‘string\’string’

test.py

```
1  print('stri\'ng')
```

Shell

```
>>> %Run test.py

  stri'ng
```

Escape Sequences

There are more types of Escape sequences

Escape-sequence	Purpose
\n	New line
\\	Backslash character
\'	Apostrophe '
\"	Quotation mark "
\a	Sound signal
\b	Slaughter (backspace key symbol)
\f	The conversion of format
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
xhh	Character with hex code hh
ooo	Character with octal value ooo
\0	Character Null (not a string terminator)
\N{*id*}	Identifier ID of Unicode database
uhhhh	16-bit Unicode character in hexadecimal format
Uhhhhhhhh	32-bit Unicode character in hexadecimal format
\другое	Not an escape sequence (\ character is stored)

Raw Strings

Syntax
r”string”

test.py

```
path = 'C:\Users\Gavin\Desktop'

print(path)
```

Shell

```
>>> %Run test.py
  Traceback (most recent call last):
    File "C:\Users\Gavin\Desktop\test.py"
  , line 1
      path = 'C:\Users\Gavin\Desktop'
           ^
  SyntaxError: (unicode error) 'unicodees
  cape' codec can't decode bytes in posit
  ion 2-3: truncated \UXXXXXXXX escape

>>>
```

test.py

```
path = r"C:\Users\Gavin\Desktop"
print(path)
```

Shell

```
>>> %Run test.py

  C:\Users\Gavin\Desktop

>>>
```

Slicing and Indexing

Syntax

string[index], string[s:e]

test.py

```python
my_string = "batman"

index1 = my_string[0]
slice1 = my_string[0:3]
slice2 = my_string[3:]

print(index1)
print(slice1)
print(slice2)
```

Shell

```
>>> %Run test.py

  b
  bat
  man
```

Length

Function
len(string)

test.py

```
1  my_string = "batman"
2  my_len = len(my_string)
3
4  print(my_len)
```

Shell

```
>>> %Run test.py

  6
```

Case

Methods

string.**upper()** etc.

test.py

```python
my_string = "batman"

upp = my_string.upper()
low = my_string.lower()
ttl = my_string.title()

print(upp)
print(low)
print(ttl)
```

Shell

```
>>> %Run test.py

  BATMAN
  batman
  Batman
```

Reverse

Syntax

string[::-1]

test.py

```python
my_hero = 'batman'

my_oreh = my_hero[::-1]

print(my_oreh)
```

Shell

```
>>> %Run test.py

  namtab
```

Find

Method

string.**find**(substring, s, e)

test.py

```python
my_string = 'where\'s wally'

there = my_string.find('wally')

print(my_string)
print(there)
```

Shell

```
>>> %Run test.py

  where's wally
  8
```

Count

Method

string.**find(**substring, s, e**)**

test.py

```python
my_string = 'where\'s wally'

there = my_string.count('w')

print(my_string)
print(there)
```

Shell

```
>>> %Run test.py

  where's wally
  2
```

Replace

Method

string.**replace(**find, rep, c**)**

test.py

```
1  my_string = 'where\'s wally'
2
3  there = my_string.replace('w', 't', 1)
4
5  print(my_string)
6  print(there)
```

Shell

```
>>> %Run test.py

  where's wally
  there's wally
```

Add

Syntax

string + string

test.py

```
str1 = 'how'
str2 = 'dare'
str3 = 'you'

str4 = str1 +' ' + str2 + ' ' + str3

print(str4)
```

Shell

```
>>> %Run test.py

  how dare you
```

Join

Method

seperator.**join**(list)

test.py

```
words = ["how", "dare", "you"]

greta = " ".join(words)

print(greta)
```

Shell

```
>>> %Run test.py

  how dare you
```

Split

Method

string.**split(**sep, c**)**

test.py

```
greta = 'how dare you'

words = greta.split(' ', 1)

print(words)
```

Shell

```
>>> %Run test.py

  ['how', 'dare you']
```

rSplit

Method

string.**rsplit(**sep, c**)**

test.py

```
greta = 'how dare you'

words = greta.rsplit(' ', 1)

print(words)
```

Shell

```
>>> %Run test.py

  ['how dare', 'you']
```

Strip

Method

string.(l/r)**strip**(sub)

test.py

```python
messy = '     string  '

clean = messy.strip()
rclean = messy.rstrip()
lclean = messy.lstrip()

print(messy  + '.')
print(clean  + '.')
print(rclean + '.')
print(lclean + '.')
```

Shell

```
>>> %Run test.py

      string  .
  string.
      string.
  string  .
```

f Strings

Syntax

f”string{var: format}”

test.py

```python
from datetime import datetime

my_today = datetime.now()

print(f"Today's date is {my_today:%B %d, %Y}")
```

Shell

```
>>> %Run test.py

  Today's date is March 26, 2020
```

Positional Formatting

Syntax

“string{var}”.**format**(var)

test.py

```
x = 'which'
y = 'formatted'

format_1 = 'this is a string {} has been positionally {}'.format(x,y)

print(format_1)
```

Shell

```
>>> %Run test.py

  this is a string which has been positionally formatted

>>> |
```

Regular Expressions

Syntax

re.match(r”regex”, string)
re.search(r”regex”, string)

test.py

```python
import re

string = '123abc'

regex = True if re.search(r"[a-z]+", string) else False

print(regex)
```

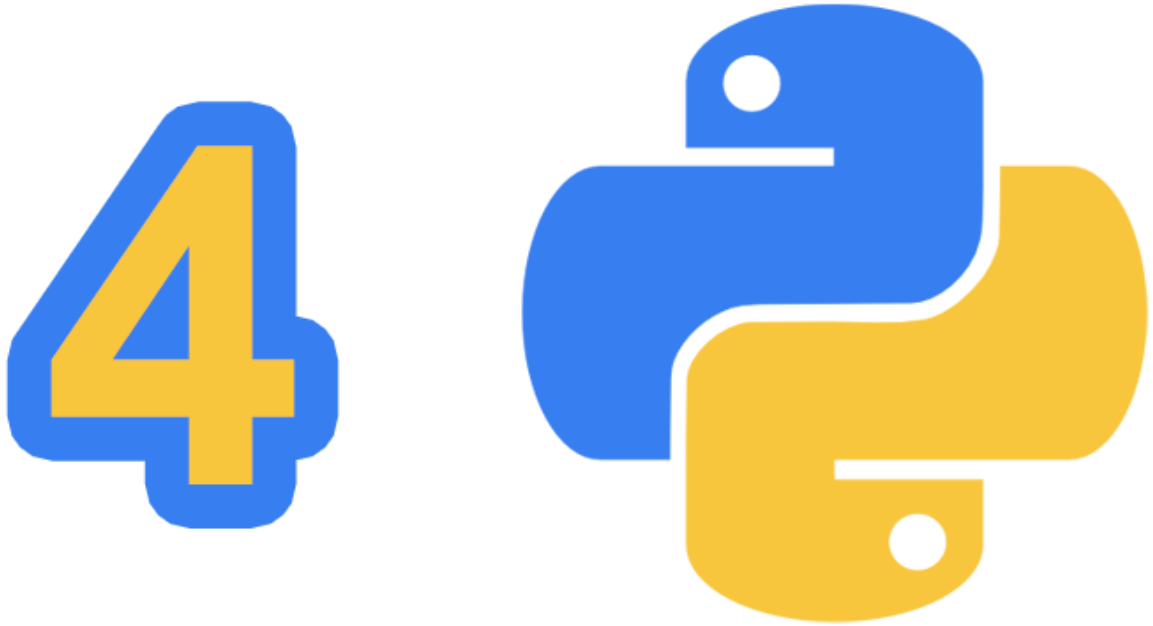
Shell

```
>>> %Run test.py

  True
```

Next on #5

Defining Functions



Python Quick Tips

Working with Strings