

CSCI 145 · Quiz 2 Reference Sheet

STRINGS — INDEXING & SLICING

Every character in a string has an index (from 0) and a negative index (from the right, starting at -1).

index	0	1	2	3	4	5	6	7
s =	C	O	M	P	U	T	E	R
neg. index	-8	-7	-6	-5	-4	-3	-2	-1

`s[i]` the single character at index `i`, e.g. `s[2]` is 'M'

`s[start:end]` a new string, the slice from start up to but **excluding** end

`s[start:]` slice from start through the end

`s[:end]` slice from beginning through (not including) end

```
s[2:5] # 'MPU'    s[:3] # 'COM'
s[5:]  # 'TER'    s[-3:] # 'TER'
```

STRINGS ARE IMMUTABLE

Once created, a string **cannot be changed in place**. Operators and methods like `+`, `*`, and `.lower()` don't modify the original — they build and return a **new** string.

```
>>> s[0] = 'c'
TypeError: 'str' object does not support item assignment
```

STRING METHODS

<code>len(s)</code>	number of characters in s
<code>s.lower()</code>	a new string, all lowercase
<code>s.upper()</code>	a new string, all uppercase
<code>s.strip()</code>	a new string, leading/trailing whitespace removed
<code>s.find(sub)</code>	index where sub first appears, or -1 if not found
<code>s.replace(old, new)</code>	a new string with every old replaced by new

WAYS TO IMPORT A MODULE

<code>import random</code>	call as <code>random.randint(...)</code>
<code>import random as r</code>	call as <code>r.randint(...)</code>
<code>from random import randint</code>	call as <code>randint(...)</code> — no prefix
<code>from random import *</code>	imports all names; call as <code>randint(...)</code>

ESCAPING SPECIAL CHARACTERS

`\` is the **escape character** in Python: `\n` newline `\t` tab `\"` or `\'` a quote that doesn't end the string `\\` a literal backslash.

```
s = "Hi!\nTab:\there. Quote: \" Back: \\"
>>> print(s)
Hi!
Tab:   here. Quote: " Back: \
```

USING A MODULE

At the top of your file, import the module. Then call its functions as `module_name.function_name(...)`. None of these functions have side effects; all of them return a value.

```
import random
```

`random.random()` a random float `x` such that $0 \leq x < 1$

`random.randint(a, b)` a random integer `x` such that $a \leq x \leq b$ (both ends included)

```
>>> random.random()
0.6964691855978616
>>> random.randint(1, 6)
4
```

LOCAL VS. GLOBAL SCOPE

- The **scope** of a variable is the region of code where it can be accessed.
- Functions create **local scope** — variables created inside a function are local to it and cannot be accessed from outside.
- Variables created outside of any function are in **global scope**, and can be accessed anywhere.

TRACING THE CALL STACK

Every function call gets its own **frame**, holding just that call's parameters and local variables. A call can happen inside another call — each one pushes a new frame onto the stack, and when it returns, that frame pops and the value flows back into the line that called it.

```
def triple(x):
    x = x * 3
    return x
def mystery(x):
    y = triple(x) + 1
    return y
```

```
x = 4
result = mystery(x)
```

main

x: 4 result: ? → 13

mystery(4)

x: 4 y → triple(4) + 1 → 12 + 1 → **13**

triple(4)

x: 4 → 12 → **12**