# TAMU ENGR-102 Exam Reference Sheet

2023 — Archer.Simmons@tamu.edu — Peer Teacher

## Special Numerical Operators

`x % y` -> Modulo: Remainder of $\frac{x}{y}$

`x // y` -> Floor Division: $\lfloor \frac{x}{y} \rfloor$

## Conditional Statements

```
if condition1:
  print("condition1 is True")
elif condition2: # Optional
  print("condition2 is True")
else: # Optional
  print("conditions 1 & 2 are False")
```

## Lists

```
Accessing Elements (x[index]):
  x = [[9, 6], 4]
  x[0][1] -> 6
  x[-2]   -> [9, 6]
  x[1]    -> 4

List/Str Slicing (x[start:end:indexJump]):
  x = ['H', 'o', 'w', 'd', 'y']
  x[2:]    -> ['w', 'd', 'y']
  x[:3]    -> ['H', 'o', 'w']
  x[2:4]   -> ['w', 'd']
  x[1:-1]  -> ['o', 'w', 'd']
  x[-5:-1] -> ['H', 'o', 'w', 'd']
  x[::-1]  -> ['y', 'd', 'w', 'o', 'H']

In-Place Methods:
  x.append(y) # Adds y to end of x
  del x[i]    # Deletes x[i]
  x.remove(y) # Deletes y
  x.pop()     # Deletes & returns x[-1]
  x.sort()    # Lexicographically sorts x
  x.index(y)  # Returns index of y

Static Methods:
  len(x) # Number of elements in x
  min(x) # Lowest number in x
  max(x) # Highest number in x
  sum(x) # Summation of numbers in x
```

## Loops

```
Keywords:
  break    # Exits loop
  continue # Skips to next iteration

While (Conditional Loop):
# Runs code until condition is False
  while condition:
    print("condition is True")

For (Ranged Loop):
  # i: Starting Value
  # n: Stopping Value
  # j: Step Value
  range(i=0, n, j=1) -> list[int][i:n:j]
  # Generates list(|{x ∈ ℤ|i ≤ x < n}|)

  x = list(range(3)) -> [0, 1, 2]
  for i in x:
    print(i) -> "0\n1\n2\n"
```

## Terminal Input/Output

```
Output to Terminal:
  # end: char at end of print
  # sep: char separating args
  print(*args, end='\n', sep=' ')

Input from Terminal:
  # Prints msg to terminal; grabs input
  inp  = input(msg)        -> str
  iinp = int(input(msg))   -> int
  finp = float(input(msg)) -> float
```

## F-Strings

```
  x, name = 3.1415, 'Ritchey'
  f"TXT | {x}"     -> "TXT | 3.1415"
  f"How-D {x:.2f}" -> "How-D 3.14"
  f"{x:10.2f}"     -> "      3.14"
  f"{name:<10}"    -> "Ritchey   "
  f"{name:>10}"    -> "   Ritchey"
  f"{name:^10}")   -> " Ritchey  "
  f"{name:*^13s}"  -> "***Ritchey***"
```

## Pythonic Quirks

```
Operator Precedence (Highest to Lowest):
  Numerical Operators (PEMDAS)
    1. ()          # Parentheses
    2. **          # Exponential
    3. -x          # Parity Negation
    4. *, /, //, % # Multiplication/Division
    5. +, -        # Addition/Subtraction
  Relational Operators
    6. ==, !=, <=, >=, # Comparisons
       >, <, is, is not
  Boolean Operators
    7. not         # Negation
    8. and         # Conjunction
    9. or          # Disjunction

Type-Casting to Boolean
  # Int & Float: ONLY 0 is False
    bool(0)   -> False
    bool(-1)  -> True
    bool(0.0) -> False

  # Str: ONLY Empty is False
    bool('') -> False

  # Lists: ONLY Empty is False
    bool([]) -> False

  # None keyword is always False
    bool(None) -> False

Type-Specific Behavior:
  Strings & Tuples are Immutable
    stringVar[0] = 'k' -> ERROR
    tupleVar[1]  = 2   -> ERROR

  List Alias Assignment
      A = [1, 2, 3]
      # Variable B is now an alias for A
      B = A
      A[1] = 27
      # Same OBJECT; not just same value
      A == B -> True
      A is B -> True
```

# TAMU ENGR-102 Exam Reference Sheet

## Special Numerical Operators:

`x % y` -> Modulo: Remainder of $\frac{x}{y}$

`x // y` -> Integer Division: $\lfloor \frac{x}{y} \rfloor$

## Conditional (if-elif-else) Statement:

```
if condition1:
    Do this
elif condition2: # Optional
    Do this instead
else: # Optional
    Otherwise, do this
```

## Lists:

```
Accessing Elements (x[index]):
x = [[9, 6], 4]
x[0][1] -> 6
x[-2]   -> [9, 6]

Slicing (x[start:end:indexJump]):
x = ['H', 'o', 'w', 'd', 'y']
x[2:]    -> ['w', 'd', 'y']
x[:3]    -> ['H', 'o', 'w']
x[2:4]   -> ['w', 'd']
x[1:-1]  -> ['o', 'w', 'd']
x[-5:-1] -> ['H', 'o', 'w', 'd']
x[::-1]  -> ['y', 'd', 'w', 'o', 'H']

Instance Methods (In-Place):
x.append(y) # Adds y to end of x
del x[i]    # Deletes x[i]
x.remove(y) # Deletes y
x.pop()     # Deletes & returns x[-1]
x.sort()    # Lexicographically sorts x
x.index(y)  # Returns index of y

Static Methods:
len(x) # Number of elements in x
min(x) # Lowest number in x
max(x) # Highest number in x
sum(x) # Summation of numbers in x
```

## Dictionaries:

```
Accessing Elements:
d = {'x':9, 'y':"Howdy", 'z':[3, 6, 1]}
d['y']     -> "Howdy"
d['z'][-2] -> 6
d.get('x') -> 9
for key in d:
    Do this for each key in d

for key, value in d.items():
    Do this for each key:value pair in d

Modification:
d[k] = v # Adds key:value pair to d
del d[k] # Deletes key:value pair
```

## Functions:

```
def func1(): # Declaration
  '''0 args; implicit return (None)'''
  Do this # Definition

def func2(a, b=25): # b has default val
  '''Positional args; explicit return'''
  Do this
  return finalVal

if __name__ == "__main__":
  func1()     -> None # Function Call
  func2(x)    -> finalVal # a = x, b = 25
  func2(x, y) -> finalVal # a = x, b = y
```

## Try-Except Block:

```
try:
    Try running this exception-prone code
except Exception: # Optional
    Run if specified exception raised
except:
    Run if any exception raised
else: # Optional
    Run only if no exception raised
finally: # Optional
    Always run this
```

## Loops:

```
Keywords:
break    # Exists loop
continue # Skips to next iteration

While:
Repeats indented code until condition is False
while condition:
    Do this

For:
Iterates over container (list, tuple, dict)
range(start, end, jump) -> list[int][::jump]
for var in range:
    Do this
```

## File IO:

```
Useful File Methods:
str.strip()          -> str # Trim edge ' ', '\n'
str.split(x)         -> list[str] # Split at x
str.join(list[str]) -> str # Joins strs in list

Open/Close File:
  File Modes:
    'r': Read starting at 0th line
    'w': Write starting at 0th line
    'a': Append starting at last line

with open(filename, mode) as file:
    Do this # Auto-closes file after indent

file = open(filename, mode)
file.close()

Read Methods:
file.read()      -> str       # All file data
file.readline()  -> str       # 1 line
file.readlines() -> list[str] # All file data

Write Methods:
file.write(str)            # Writes str
file.writelines(list[str]) # Writes list[str]
```