CS110 PYTHON QUICK REFERENCE GUIDE

Under Development: Contains Content up to Assessment #1

USAF Academy Department of Computer Science

Useful Modules (add using the 'import' command)

math – Advanced Math Functions	**random** – Random Numbers
pythonGraph – Graphics	

Output — Lesson 2

```
print('hello world')    # Text Only
>>> hello world↵

print(my_variable)      # Variables Only
>>> 12345↵
    (assuming my_variable = 12345)

print('hello', name)    # Combining text/vars
>>> hello Bob↵
    (assuming name = 'Bob')

print(my_variable + 1) # Math Expression
>>> 12346↵
    (assuming my_variable = 12345)

print('hello', end='') # Changing line end
>>> hello world
```

String Concatenation

```
print('You are ' + str(age) + 'years old')
(Use str() to convert non-strings to strings)
```

Formatted Printing
(%d = integer, %f = float, %s = string)

```
print('hi %d %0.2f %s' % (1, 2.3456, 'Bob'))
>>> hello 1 2.35 Bob
```

The 2 here tells Python to only print/round to 2 decimal places.

Special Characters

```
\n = New Line
\t = Tab
\\ = the '\' character
```

Options

```
end='\n' #Last character
sep=' ' #Separator between commas
```

Input — Lesson 2

```
variable_name = input('optional prompt goes here: ')
>>> optional prompt goes here: user typed value goes here
```

NOTE: `input()` will always return a string. To convert to another data type, use the following functions:

```
- To convert to integer:     int(value) -OR- int(input())
- To convert to float:       float(value) -OR- float(input())
- To convert to a character: chr(value) -OR- chr(input())
- To convert to a string:    str(value) -OR- input()
```

Assignment — Lesson 2

```
       Stored In
variable_name = expression

x = 1
y = input()
my_variable = a + b + 23
```

Variables must start with letters, but can contain numbers and _.
Variable names are case sensitive

Basic Math — Lesson 2

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus (remainder)	%
Exponent	**

Order: Parentheses, Exponents, Multiply/Divide, Add/Subtract

Advanced Math — Lesson 3

Built-In Functions

Operation	Example	Returns
Absolute Value	abs(-3.2)	3.2
Rounding	round(3.57,1)	3.6
Power	pow(4,3)	64

Included in the Math Module

math.pi	math.e
math.sin(VALUE)	math.ceil(VALUE)
math.cos(VALUE)	math.floor(VALUE)
math.tan(VALUE)	math.sqrt(VALUE)

Random Numbers — Lesson 6

Included in the Random Module

```
x = random.random()          # x assigned a random float; 0.0 <= x < 1.0
x = random.randint(min, max) # x assigned a random int; min <= x <= max
```

Conditional Logic — Lesson 4

```
if LOGICAL TEST(S):
    DO WHEN ABOVE IS TRUE          } Required
elif LOGICAL TEST(S):
    DO WHEN ABOVE IS TRUE          } Repeat as Needed
. . .                              } Optional
else:
    DO IF NOTHING IS TRUE
```

Only the first TRUE block of code will execute.

Comparison Operator	Symbol	Example
Equal To	==	name == 'bob'
Not Equal To	!=	x != 5
Greater Than	>	2023 > 2006
Greater Than or Equal To	>=	gpa >= 2.0
Less Than	<	21 < your age
Less Than or Equal To	<=	x

Logical Operator	Description	Example
and	True if BOTH conditions are True	GPA >= 3.0 and GPA <= 4.0
or	True if either condition is True	GPA < 2.0 or PEA < 2.0 or MPA < 2.0
not	True if the condition is not True	not (GPA < 2.0)

Loops / Iteration — Lesson 7-8

Loops allow a group of statements to be executed multiple times.

While Loop

Continues executing the same sequence of code as long as the test condition evaluates to **True**.

Ex #1) Using a while loop to count from 0 - 9

```
i = 0              Initialize the loop control variable

while i < 10:      Test the loop control variable

    # Code here will
    # execute 10x          Execute Statements

    i = i + 1      Modify Loop Control Variable
```

Ex #2) Input Validation (keep getting values from the user until the user types -1)

```
user_input = 0     Initialize the loop control variable

while user_input != -1:  Test

    # Gets the Value from the User
    user_input = int(input('Value:'))  Modify

    # (Optional) Executes Code
    Execute
```

This loop will repeat until the user provides the required value (*i.e.*, -1, in this case).
Here, user_input is the loop control variable.

For Loop

Repeats a predetermined number of times, or iterates over a sequence (e.g., a list).

For loops are useful when you know in advance how many times the loop needs to execute.

Ex #1) Using a for loop to count from 0 - 9

```
for i in range(0, 10):   Initialize, Test, and
                         Modify occurs here.
    # Code here will
    # execute 10x        Execute Statements
```

Ex #2) Looping over all elements in a list

```
my_list = ['a', 'b', 'c', 'd']  Initialize List

for list_element in my_list:

    # Code Goes Here
    print(list_element)
```

In this example, list_element will have 'a' stored in it during the first iteration, 'b' during the second, etc.

Program Output:

```
a
b
c
d
```

Every For Loop can be coded using a While Loop, but not every While Loop can be coded using a For Loop.
Only use For Loops when you know how many times the loop needs to execute.

pythonGraph

Static Drawing Template

Use when drawing a simple, non-animated picture.

```
import pythonGraph

# Setup tasks for window
pythonGraph.open_window(640, 480)
pythonGraph.set_window_title("pythonGraph")

# Custom Code Goes Here

# Wait using the window is closed
pythonGraph.wait_for_close()
```

Drawing Methods

```
clear_window(color)
draw_arc(x1, y1, x2, y2, start_x, start_y, end_x, end_y, color, width)
draw_image(filename, x, y, width, height)
draw_rectangle(x1, y1, x2, y2, color, filled, width)
draw_circle(x, y, radius, color, filled, width)
draw_ellipse(x1, y1, x2, y2, color, filled, width)
draw_line(x1, y1, x2, y2, color, width)
draw_pixel(x, y, color)
draw_text(text, x, y, color, font_size)
```

- **Bolded** items indicate optional parameters
- Filled should be set to **True** or **False**

Supported Colors

```
pythonGraph.colors.
  BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN,
  LIGHT_GRAY, DARK_GRAY, LIGHT_BLUE, LIGHT_GREEN,
  LIGHT_CYAN, LIGHT_RED, LIGHT_MAGENTA, YELLOW, WHITE
```

Mouse Functions

```
get_mouse_x() → returns x coordinate
get_mouse_y() → returns y coordinate
mouse_button_pressed(which_button)
which_button = pythonGraph.mouse_buttons.LEFT/CENTER/RIGHT
```

Keyboard Functions

```
mouse_button_pressed(which_key)
which_key = 'a', 'f1', 'up', 'escape', . . .
```