# Cheat Sheet

### while loop (indefinite loops; don't know in advance how many times it will repeat)

|  |  |
| --- | --- |
| Relational/Logical Operators | |
| Operator | Description |
| < | less than |
| <= | less than or equal |
| > | greater than |
| >= | greater or equal |
| == | equal |
| != | not equal |
| and | and |
| or | or |
| not | not |

while condition:

statement(s)

Example:

number = 1

**while number <= 200:**

print(str(number) + " ")

number = number \* 2

### boolean (logical values of True or False)

iLoveCS = True

minor = (age < 21)

if **minor**:

print("Can't purchase alcohol!")

def both\_odd(n1, n2):

return **(n1 % 2 != 0 and n2 % 2 != 0)** # returns True or False

|  |  |
| --- | --- |
| Function | Description |
| random() | returns a random float in the range [0, 1)  in other words, 0 inclusive to max exclusive |
| randint(min, max) | returns a random integer in the range [min, max) in other words, min to max-1 inclusive |

### Random (produce random numbers)

import random

# ranom number 1 - 10

rand = random.**randint(1, 11)**

### String (store text)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| char | P | . |  | D | i | d | d | y |

name = "P. Diddy"

print(len(name)) # 8

|  |  |
| --- | --- |
| Function | Description |
| find(str) | index where the start of the given string appears in this string (-1 if not found) |
| lower() | a new string with all lowercase letters |
| upper() | a new string with all uppercase letters |

### Fencepost Loop (N posts, N-1 wires between)

def print\_numbers(max):

**print(1, end=’’)**

for i in range(2, max + 1):

print(", " + str(i), end=’’)

print() # end the line

print\_numbers(5) outputs 1, 2, 3, 4, 5

# Questions

### While Loop Mystery

1. Consider the following function. For each call below, indicate what output is produced.

|  |  |
| --- | --- |
| def mystery1(x):  y = 1  z = 0  while 2 \* y <= x:  y = y \* 2  z += 1    print(y, z) | Call Output  mystery1(1) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery1(6) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery1(19) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery1(39) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery1(74) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

2. Consider the following function. For each call below, indicate what output is produced.

|  |  |
| --- | --- |
| def mystery2(x):  y = 0  while x % 2 == 0:  y += 1  x = x / 2    println(x, y) | Call Output  mystery2(19) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery2(42) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery2(48) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery2(40) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery2(64) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

3. Consider the following function. For each call below, indicate what value is returned.

|  |  |
| --- | --- |
| def mystery3(x, y):  while x != 0 and y != 0:  if x < y:  y = y - x  else:  x = x - y    return x + y | Call Value Returned  mystery3(3, 3) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery3(5, 3) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery3(2, 6) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery3(12, 18) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  mystery3(30, 75) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

### While Loop Programming

4. Write a function show\_twos that shows the factors of 2 in an integer. For example:

**Call** **Output**

show\_twos(7) 7 = 7

show\_twos(18) 18 = 2 \* 9

show\_twos(68) 68 = 2 \* 2 \* 17

show\_twos(120) 120 = 2 \* 2 \* 2 \* 15

The idea is to express the number as a product of factors of 2 and an odd number. The number 120 has 3 factors of 2 multiplied by the odd number 15. For odd numbers (e.g. 7), there are no factors of 2, so you just show the number itself. Assume that your function is passed a number greater than 0.

# Questions (continued)

### While Loop Programming

5. Write a function show\_hailstone that takes an integer parameter *n* and that displays the hailstone sequence starting at n and ending with 1.  In a hailstone sequence, each value x is followed either by:  
  
        3*x* + 1 *if x is odd*  
        *x*/2 *if x is even*  
  
Below are a series of calls and the output produced:

**Call** **Output**

show\_hailstone(3)     sequence for 3: 3, 10, 5, 16, 8, 4, 2, 1  
show\_hailstone(10)     sequence for 10: 10, 5, 16, 8, 4, 2, 1  
show\_hailstone(1)     sequence for 1: 1

It is believed that for any positive integer n, the sequence always reaches 1, although nobody has yet proven that this is true.  Assume that your function is passed a number greater than 0.

### Boolean Logic

6. Write a function named sign that accepts two integers as parameters, and that returns a String indicating the sign of the result of multiplying the integers together. Your function should either return "Positive", "Negative", or "Zero". Do not perform any arithmetic operations on the integers (+, -, \*, /). Example calls and return values are listed below.

**Call** **Return Value**

sign(4, 0)     "Zero"  
sign(2, -3)     "Negative"

### Random Numbers

7. Write a function named roll\_six that simulates the repeated rolling of one six-sided die until a six is rolled. You should use random to give an equal chance of rolling a one through six. Each time the die is rolled, you should display the number seen. When you roll a six, you should print the number of trials taken. An example output of a call to roll\_six is shown below:

Rolled: 3

Rolled: 1

Rolled: 6

You got a six in 3 turns!

8. Write a function named random\_walk that performs a random one-dimensional walk, reporting each position reached and the maximum position reached during the walk. The random walk should begin at position 0. On each step, you should either increase or decrease the position by 1 (with equal probability). The walk stops when 3 or -3 is hit. The output should look like this:

position = 0

position = 1

position = 0

position = -1

position = -2

position = -1

position = -2

position = -3

max position = 1

# Solutions

1.

|  |  |
| --- | --- |
| Call | Output |
| mystery1(1)  mystery1(6)  mystery1(19)  mystery1(39)  mystery1(74) | 1 0  4 2  16 4  32 5  64 6 |

2.

|  |  |
| --- | --- |
| Call | Output |
| mystery2(19)  mystery2(42)  mystery2(48)  mystery2(40)  mystery2(64) | 19 0  21 1  3 4  5 3  1 6 |

3.

|  |  |
| --- | --- |
| Call | Value Returned |
| mystery3(3, 3)  mystery3(5, 3)  mystery3(2, 6)  mystery3(12, 18)  mystery3(30, 75) | 3  1  2  6  15 |

4.

def show\_twos(n):

print(str(n) + " = ")

while n % 2 == 0:

print("2 \* ", end=’’)

n = n / 2

print(n)

5.

def show\_hailstone(n):   
    print("sequence for " + str(n) + ": " + str(n), end=’’)   
    while n != 1:  
        if n % 2 == 0:   
            n /= 2   
        else:   
            n = 3 \* n + 1   
          
        print(", ", n, end=’’)   
      
    print()

6.

def sign(a, b):

if a == 0 or b == 0:

return "Zero"

elif (a < 0 and b < 0) or (a > 0 and b > 0):

return "Positive"

else: # (a < 0 and b > 0) or (a > 0 and b < 0)

return "Negative"

**Solutions (continued)**

7.

def roll\_six():

roll = randint(1, 6)

count = 1

print("Rolled:", roll)

while roll != 6:

roll = random.randint(1, 6)

print("Rolled:", roll)

count += 1

print("You got a six in", count, "turns.")

8.

def random\_walk():

n = 0

maximum = 0

print("position =", n)

while -3 < n and n < 3:

flip = random.randint(0, 2)

if flip == 0:

n += 1

else:

n -= 1

maximum = max(n, maximum)

print("position =", n)

print("max position =", maximum)