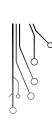


CONDITIONAL ITERATION

COMP130 – INTRODUCTION TO COMPUTING DICKINSON COLLEGE



THE WHILE LOOP

- The while loop provides conditional repetition.
 - When program execution reaches the loop header the loop condition is checked. As long
 as (i.e. while) the loop condition is True the statements in the loop body are executed
 over and over again.

```
magic_number=random.randint(1,5) Condition

Guess = int(input("What is your guess?"))

while guess != magic_number:

guess = int(input("Nope! Guess again: "))

print("You got it!!!")
```

WHILE LOOP IDIOM

- Setup: Variables used in the loop condition are initialized before the loop header.
- Loop Condition: Checked before each execution of the loop body.
- Update: Variables used in the loop condition are modified each time the loop body is executed. This allows the loop to terminate.

```
init_investment = float(input('What is the initial investment? '))
interest_rate = float(input('What is the initial investment? '))

current_value = init_investment
compoundings = 0

while current_value < (init_investment*2)

interest = current_value * interest_rate
current_value = current_value + interest

compoundings = compoundings + 1

print('The investment doubled in ' + str(compoundings) + ' compoundings.')</pre>
```



THE BREAK STATEMENT

 The break statement terminates the execution of a loop (i.e. it breaks out of the loop). When a break is executed, the program execution continues following the loop body.







FLOATING POINT ROUNDING ERRORS

• float values are often approximations of real values, which leads to rounding errors. Thus, care is required when comparing float values.

```
x = 1/5
y = 3/5
z = x + x + x

equal = (y == z)  # not exactly equal...
print(equal)

epsilon = 0.00000001
equal = abs(y-z) < epsilon # but close enough...
print(equal)</pre>
```