• Python Boolean/conditionals/relational operators, cont’d

• Repetition (Looping) in Python

Days of programming can save you hours of planning.

a relational operator for sequences: in

>>> 'a' in 'abc'
True
>>> 'bc' in 'abc'
True
>>> 'cde' in 'abc'
False
>> 'a' < 'b'
True

>>> 'apple' < 'banana'
True

>>> 'twenty-five' > 'twenty'
True

endswith, startswith, islower, isupper, isalnum

>>> ‘Hello’.isupper()  >>> str.isupper(‘Hello’)
False

>>> ‘hello’.islower()  >>> str.islower(‘hello’)
True
Be careful!

```python
>>> 'Apple' < 'banana'
True
>>> 'apple' < 'Banana'
False
>>> 'Zebra' < 'banana'
True

>>> x = 100.0; y = 100; x == y
True
>>> x = 100.0; y = 100.0000000000000001; x == y
True
>>> x = 1/3; y = .33333333; x == y
False
>>> x = 1/3; y = .3333333333333333; x == y
True
```
def check(n):
    '''what does this function do?'''
    if (n % 2) == 0:
        return True
    else:
        return False

>>> check(100)
>>> check(101)
def checkfit(s1, s2):
    """(str, str) -> Boolean """

    >>> checkfit('Deschutes', 'Metolius')  # 17
    True
    >>> checkfit('Mississippi', 'Missouri')  # 19
    False
    
    field_width = 18
    ttl_len = len(s1) + len(s2)
    fits = ttl_len <= field_width

    return fits  # return Boolean

    >>> checkfit('Deschutes', 'Metolius')

    def checkfit(s1, s2):
        """(str, str) -> int """

        >>> checkfit('Deschutes', 'Metolius')  # 17
        True
        >>> checkfit('Mississippi', 'Missouri')  # 19
        False
        
        field_width = 18
        padding = 10
        ttl_len = len(s1) + len(s2)
        fits = ttl_len <= field_width

        if fits:
            space = ttl_len + padding
        else:
            space = ttl_len + (padding // 2)

        return space

        >>> checklen('Deschutes', 'Metolius')
Write a function that implements the game Rock, Paper, Scissors. Each of two players is asked for their selection, and the function prints out which player wins. None value is returned.

```python
>>> rps()
Side 1 is r and Side 2 is p.
Side 2 wins!
```

```python
def rps():
    ''' '''
    s1 = random.choice('rps')
    s2 = random.choice('rps')
    if s1 == s2:
        winner = 'tie'
    elif (s1 == 'r') and (s2 == 'p'):
        winner = 'side2'
    elif (s1 == 'r') and (s2 == 's'):
        winner = 'side1'
    elif (s1 == 'p') and (s2 == 'r'):
        winner = 'side1'
    ...
    else:
        winner = 'bug'
    print('Side1 is {} and Side2 is {}'.format(s1, s2))
    print('The winner is {}'.format(winner))
    return None

>>> rps()
```
response = input('Play rps? (y or n): ')  
while response == 'y':  
    [rps code]  

while <boolean expression>:  
    <block of code>  

    while loop (indefinite loop)  
    runs while a condition is True  
    most general kind of loop

---

Example

Calculate the growth of bacteria using a simple growth model:

\[ b_{t+1} = b_t + rb_t \]

where \( b_t \) is the number of bacteria at time \( t \), and \( r \) is the growth rate (percent).

How long does it take to double the original number of bacteria?
def bacteria(population, rate):
   """(number, number) -> int

   Return time to double population of bacteria growing at percent rate per minute.
   b(t + 1) = b(t) + rb(t)

   >>> bacteria(1000, 21)
   4
   ""
   t = 0
   orig_pop = population
   prate = rate * .01

   while population < (2 * orig_pop):
       population = population + (population * prate)
       t += 1
   return t

Check list for indefinite loops

1. Set up the loop end condition.
2. Initialize the loop/counter variable (outside of the loop).
3. Write the body of the loop.
4. Advance the loop/counter variable toward the end condition.
def bacteria(population, rate):
    '''(number, number) -> int
    
    Return time to double population of bacteria growing at percent rate per minute.
    b(t + 1) = b(t) + rb(t)
    
    >>> bacteria(1000, 21)
    4
    '''
    t = 0
    orig_pop = population
    ratep = rate * .01
    
    while population < (2 * orig_pop):
        population = population + (population * ratep)
        t += 1
    return t

What is the value of x after the following Python code is executed?

x = 3  # initialize loop variable
while (x < 100):  # determine end condition
    print(x)  # body of loop, including
    x = x + 10  # advance loop variable
    # toward end condition
<whatever happens next>  # outside of block
What is the value of x after the following Python code is executed?

```python
x = 3  # initialize loop variable
while (x != 100):  # determine end condition
    print(x)  # body of loop, including
    x = x + 10  # advance loop variable
    # toward end condition
<whatever happens next>  # outside of block
```

Example

Vowel finder: Write a function, `find_vowels`, that has one parameter of type string, and returns a string of the vowels ('a', 'e', 'i', 'o', 'u') that appear in the input string. For example,

```python
>>> find_vowels('The quick brown fox')
'euiuo'
```
example
   >>> find_vowels('The quick brown fox')
   'euioo'

algorithm (string -> string)
1. check the next character in string argument
2. if it is a vowel, add it to the result string
3. repeat until reach the end of the string argument

Python code?

```python
def vowel_finder(astring):
    '''(str) -> str

    >>> vowel_finder('The quick brown fox')
    'euioo'
    '''
    vowels = 'aeiou'
    vowelstring = ''
    ctr = 0 # initialize the loop counter
    astrlen = len(astring)

    while ctr < astrlen: # what is the loop end condition?
        nextchar = astring[ctr]
        if nextchar in vowels: # body of the loop
            vowelstring += nextchar
            ctr += 1 # advance the loop counter
    return vowelstring
```
Python for is a special case of a while loop:

```python
for <var> in <sequence>:
    <body>
```

use for definite iteration
(when we know in advance how many times the loop code should run)

for example,
look at each character in a string
draw a square
CIS 122 Introduction to Programming and Problem Solving  Spring 2017

vowels = 'aeiou'
vowelstring = ''
ctr = 0  # initialize counter
astrlen = len(astring)

while ctr < astrlen:  # set end condition
    nextchar = astring[ctr]
    if nextchar in vowels:  # body of the loop
        vowelstring += nextchar
    ctr += 1  # advance counter

for nextchar in astring:  # takes care of setup/end
    if nextchar in vowels:  # body of the loop
        vowelstring += nextchar

CIS 122 Introduction to Programming Spring 2017

s = 'abcd'
count = 0
newstr = ''
for ch in s:
    newstr += s[count] * 2
    count += 1
print(newstr)
```python
CIS 122 Introduction to Programming Spring 2017

s = 'abcd'  s = 'abcd'
count = 0
newstr = ''  newstr = ''
for ch in s:
    newstr += s[count] * 2  newstr += ch * 2
count += 1
print(newstr)  print(newstr)

CIS 122 Introduction to Programming Spring 2017

checkdigit = [from code above]

if checkdigit == 10:
    checkdigit = ‘X’

if checkdigit == 11:
    checkdigit = ‘0’

isbn = isbn + str(checkdigit)

>>> isbn_gendigit('020141452')  #020141452X
02014145210
```
Python toolkit, 1.
Data types - operators and built in functions

• numeric -
  +, -, *, /, //, %, **, <, >, <=, >=, !=, abs, pow, round, int, float, min, max

• string –
  +, *, <, >, <=, >=, !=, len, input
  string methods - count, find, index, isupper, islower, isalnum, endswith, startswith, upper, lower, ...

• Boolean - not, and, or

• sequential data types (e.g., string): min, max, [], :, for

• multiple data types – help, id, print, int, float, str

Python toolkit, 2.
Other data types -
• functions (executable)
  • None

variable assignment
expression evaluation

import
modules – math, turtle, random
automated testing – doctest module

function definition (def)
function docstrings (type contract, description, examples of use)
executing (calling) a function

conditionals/selection (if elif else)
loops (while and for)
Summary of Topics for Midterm 2

- **Python toolkit – tools and use**
- **User-defined functions:**
  - program/function design recipe
  - function header and docstring
  - parameters, arguments, local vars
  - functions are expressions; they evaluate to a value
  - return statement, returned values, None
  - what happens when a function is called
  - function namespace
- **Conditionals/selection (if elif else)**
- **Loops (for while)**

---

- **Programming = CT + Coding**
- **High level languages:** Python, etc.
- **Python primitive elements**
  - Objects (type/value(s)/id)
- **Type – values and operations**
- **Primitive elements can be combined**
- **Expressions evaluate to a value**
- **Assignment statements (are not expressions)**
  - associate name/object
- **Variables are expressions**
- **Order of operations**
CIS 122 Introduction to Programming and Problem Solving  Spring 2017

• Syntax, Runtime, and Logic Errors
• Good programming style (whitespace, variable names, docstrings)

Errors

• Syntax - program won’t run, for example
  >>> def myfunc(a, b)
      print(‘hello, world’)

• Runtime - program runs until it can’t, for example
  >>> value = true

• Logical - program runs but result is incorrect
  >>> def add1(x):
      return x + 2
  >>> add1(99)
Errors

Other runtime errors

NameError
TypeError
SyntaxError
IndexOutOfRangeError
DivisionbyZeroError

Be alert

• assignment statement v. expression evaluation
• defining and calling functions
• parameters and arguments
• function returned values and side effects
• functions always return a value (it may be None)
• variables v. strings v. Boolean values
• parameters v. local variables v. input
• flow of control – watch blocks of code
Return and Print (Side Effects)

1. Functions always return a value
2. The value may be None
3. Functions sometimes have a side effect, for example, print

<table>
<thead>
<tr>
<th>Return</th>
<th>Print/Side Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes/None</td>
<td>yes</td>
</tr>
<tr>
<td>yes/value</td>
<td>no</td>
</tr>
<tr>
<td>yes/None</td>
<td>no</td>
</tr>
<tr>
<td>yes/value</td>
<td>yes</td>
</tr>
</tbody>
</table>

```
def check1():
    s = 'hello, world'
    print(s)
    return None

def check2():
    s = 'hello, world'
    return s

def check3():
    s = 'hello, world'
    return None

def check4():
    s = 'hello, world'
    print(s)
    return s
```
Thinking:

- example(s) of use
- type contract
- brief description
- algorithm

Turn it into Python code:

- write docstring (+ pass and return None)
- review Python toolkit
- write Python code

Test code using examples
Revise and retest