SILVER OAK UNIVERSITY

EDUCATION TO INNOVATION

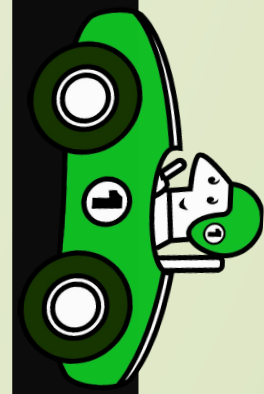
SILVER OAK COLLEGE OF COMPUTER APPLICATION

SUBJECT :Programming in Python

TOPIC : Python Flow control , Loops & Strings

Conditional Statements

- In daily routine
  - If it is very hot, I will skip exercise.
  - If there is a quiz tomorrow, I will first study and then sleep. Otherwise I will sleep now.
  - If I have to buy coffee, I will go left. Else I will go straight



if-else statement

- Compare two integers and print the min.

```
if  x < y:
        print (x)
else:
        print (y)
print ('is the minimum')
```

1. Check if x is less than y.
2. If so, print x
3. Otherwise, print y.

Indentation

- Indentation is **important** in Python
  - grouping of statement (block of statements)
  - no explicit brackets, e.g. { }, to group statements

```
x,y = 6,10

if x < y:
    print (x)
else:
    print (y)
    print('e min')
```

skipped

Run the program

6

10

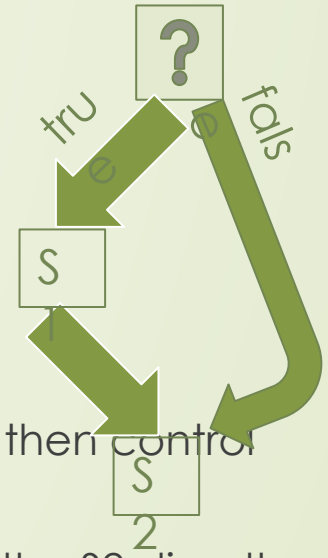
Output

6

if statement (no else!)

- General form of the if statement

```
if boolean-expr :
        S1
S2
```

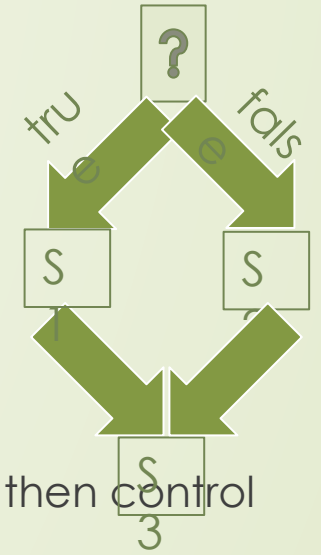


- Exec
  - Fir
  - If it evaluates to a **true** value, then S1 is executed and then control moves to the S2.
  - If expression evaluates to **false**, then control moves to the S2 directly.

if-else statement

- General form of the if-else statement

```
if boolean-expr :
        S1
else:
        S2
S3
```



- Execu
  - Fi
  - If it evaluates to a **true** value, then S1 is executed and then control moves to S3.
  - If expression evaluates to **false**, then S2 is executed and then control moves to S3.
  - S1/S2 can be **blocks** of statements!

Nested if, if-else

```
if a <= b:
        if a <= c:
            …
        else:
            …
else:
        if b <= c) :
            …
        else:
            …
```

Elif

- A special kind of nesting is the chain of if-else-if-else-… statements
- Can be written elegantly using if-elif-..-else

```
if cond1:
    s1
else:
    if cond2:
        s2
    else:
        if cond3:
            s3
        else:
            …
```

```
if cond1:
    s1
elif cond2:
    s2
elif cond3:
    s3
elif  …
else
    last-block-of-stmt
```

Summary of if, if-else

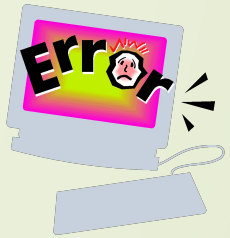
- if-else, nested if's, elif.
- Multiple ways to solve a problem
  - issues of readability, maintainability
  - and efficiency

Class Quiz

- What is the value of expression:

(5<2) and (3/0 > 1)

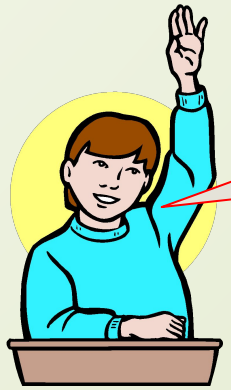
a) Run time crash/error



b) I don't know / I don't care

c) False

d) True



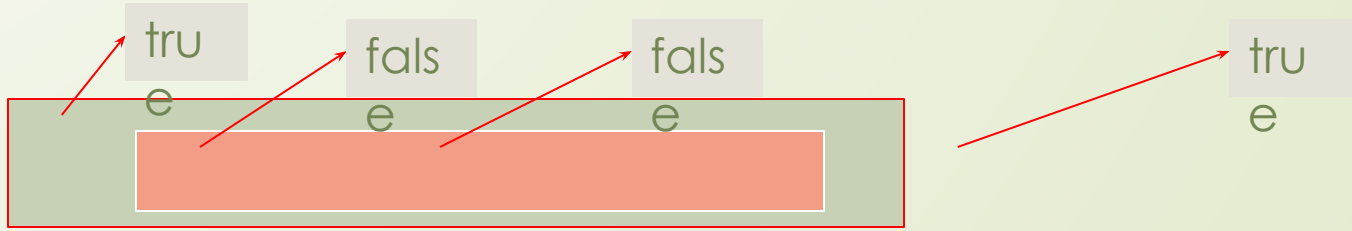
The correct answer is **False**

Short-circuit Evaluation

- Do not evaluate the second operand of binary short-circuit logical operator if the result can be deduced from the first operand
  - Also applies to nested logical operators

not((2>5) and (3/0 > 1)) or (4/0 < 2)

Evaluates to true



3 Factors for Expr Evaluation

- **Precedence**
  - Applied to two different class of operators
  - + and *, - and *, and and or, …
- **Associativity**
  - Applied to operators of same class
  - * and *, + and -, * and /, …
- **Order**
  - Precedence and associativity **identify the operands** for each operator
  - **Not which operand is evaluated first**
  - Python evaluates expressions from left to right
  - While evaluating an assignment, the right-hand side is evaluated before the left-hand side.

Class Quiz

- What is the output of the following program:

```
y = 0.1*3
if y != 0.3:
    print ('Launch a Missile')
else:
    print ("Let's have peace")
```

Launch a Missile

Caution about Using Floats

- Representation of *real numbers* in a computer can not be exact
  - Computers have limited memory to store data
    - *Between any two distinct real numbers, there are infinitely many real numbers.*
  - On a typical machine running Python, there are 53 bits of precision available for a Python float

Caution about Using Floats

- The value stored internally for the decimal number 0.1 is the binary fraction

0.0001100110011001100110011001100110011001100110011001100110011010

- Equivalent to decimal value

0.1000000000000000055511151231257827021181583404541015625

- Approximation is similar to decimal approximation 1/3 = 0.333333333...
- No matter how many digits you use, you have an approximation

Comparing Floats

- Because of the approximations, comparison of floats is not exact.
- **Solution?**
- Instead of

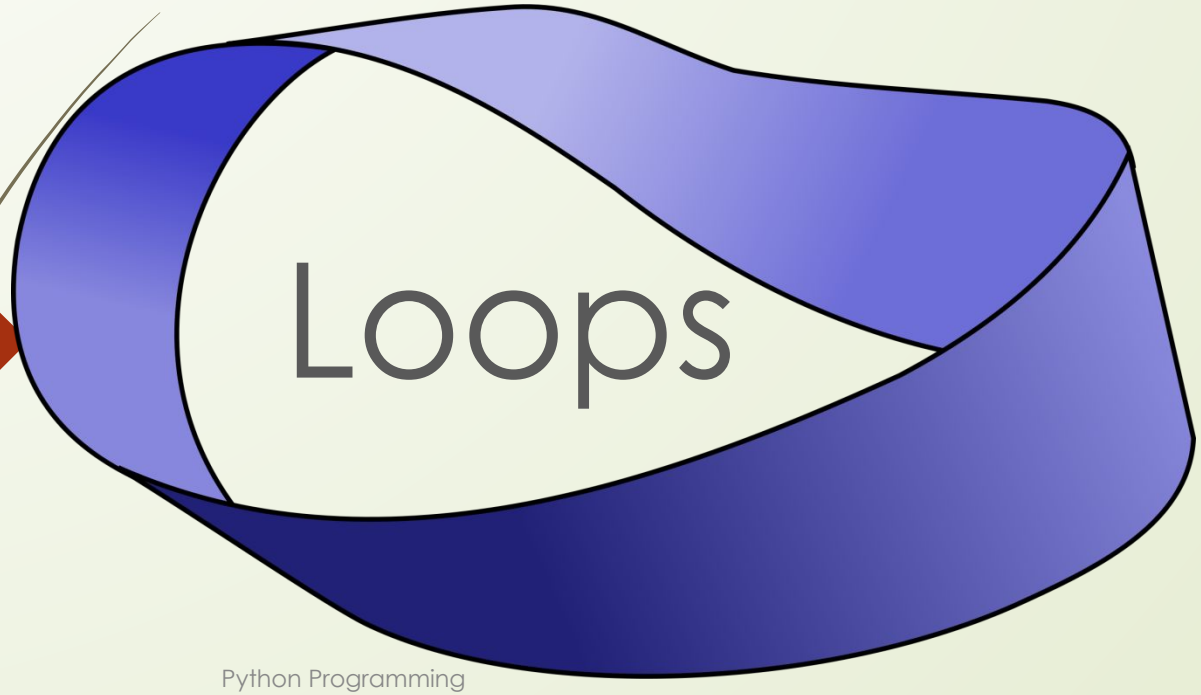
x == y

use

abs(x-y) <= epsilon

where epsilon is a suitably chosen small value

Programming using Python

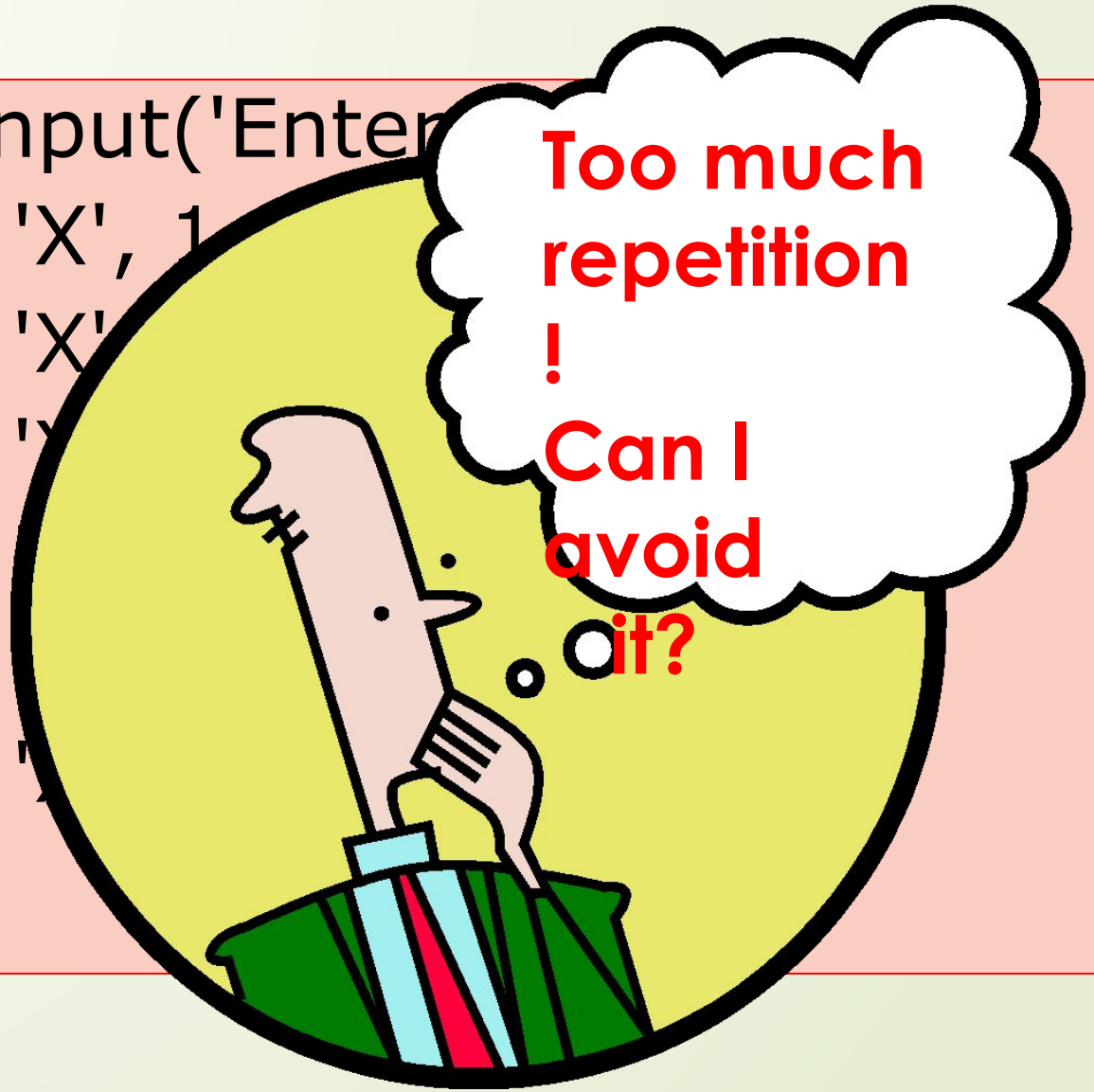


Printing Multiplication Table

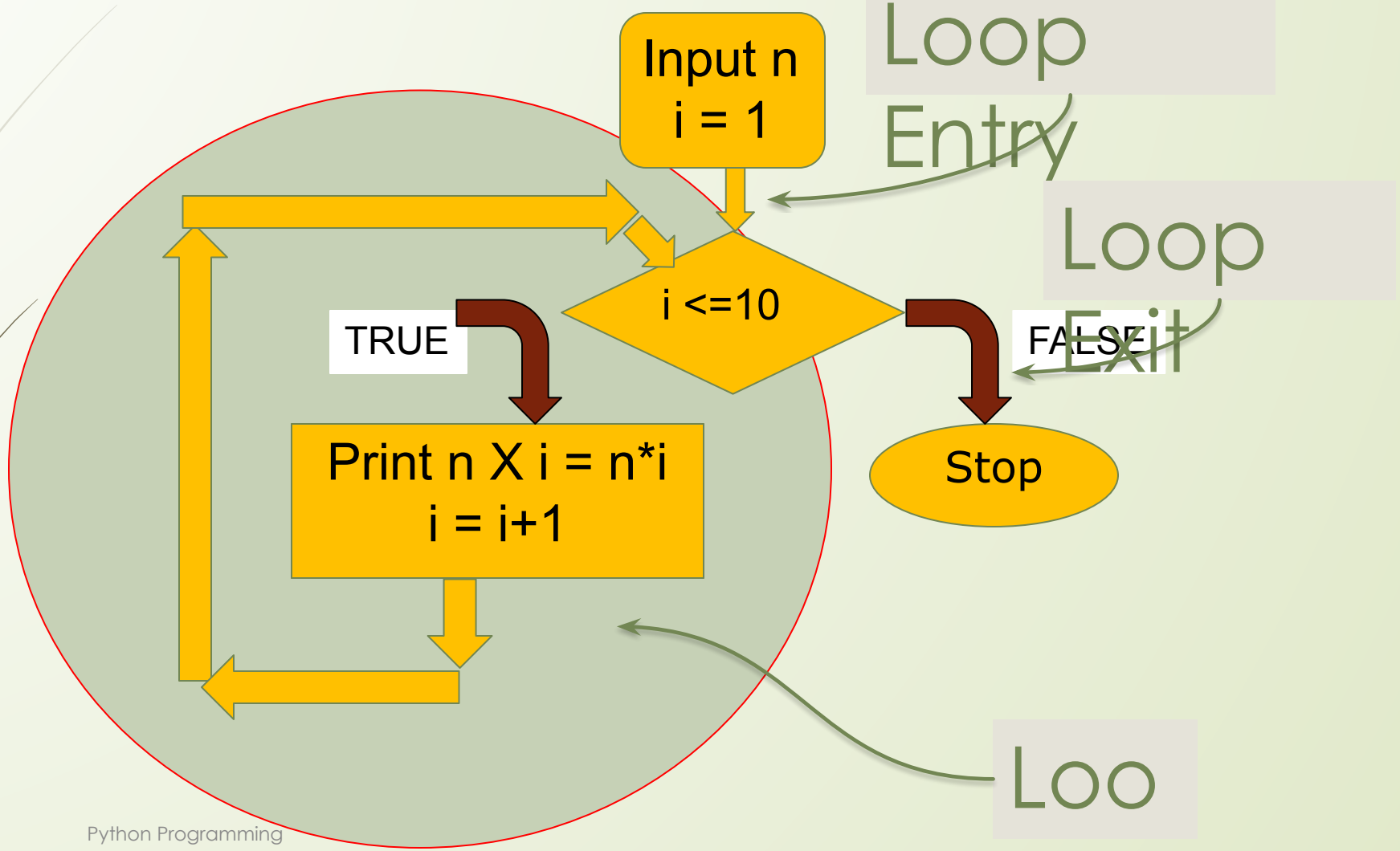
5	X	1	=	5
5	X	2	=	10
5	X	3	=	15
5	X	4	=	20
5	X	5	=	25
5	X	6	=	30
5	X	7	=	35
5	X	8	=	40
5	X	9	=	45
5	X	10	=	50

Program…

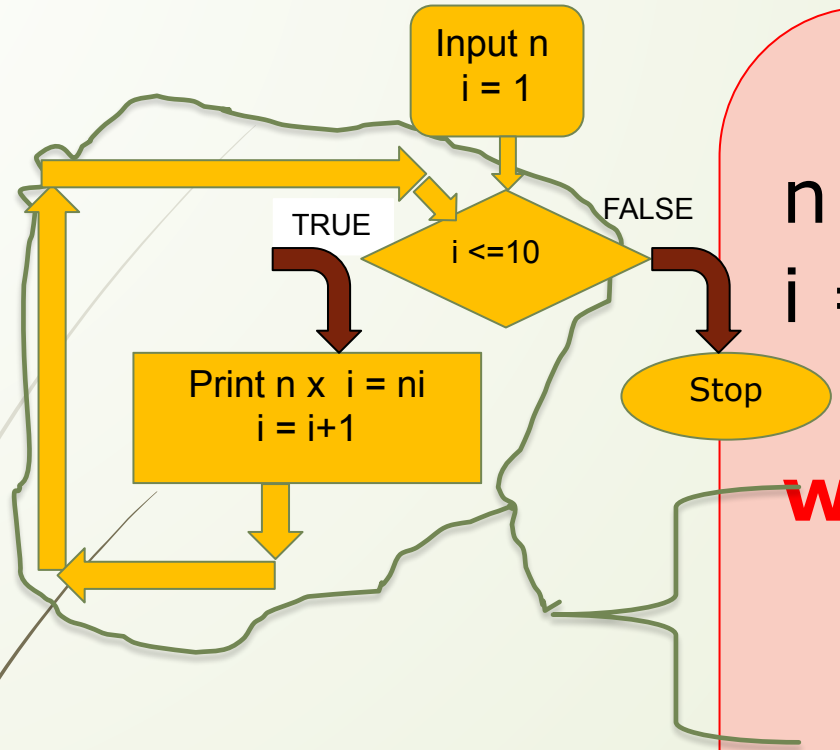

```
n = int(input('Enter
print (n, 'X', 1
print (n, 'X'
print (n, '
print (n,
print (n,
print (n, '
....
```



Printing Multiplication Table



Printing Multiplication Table

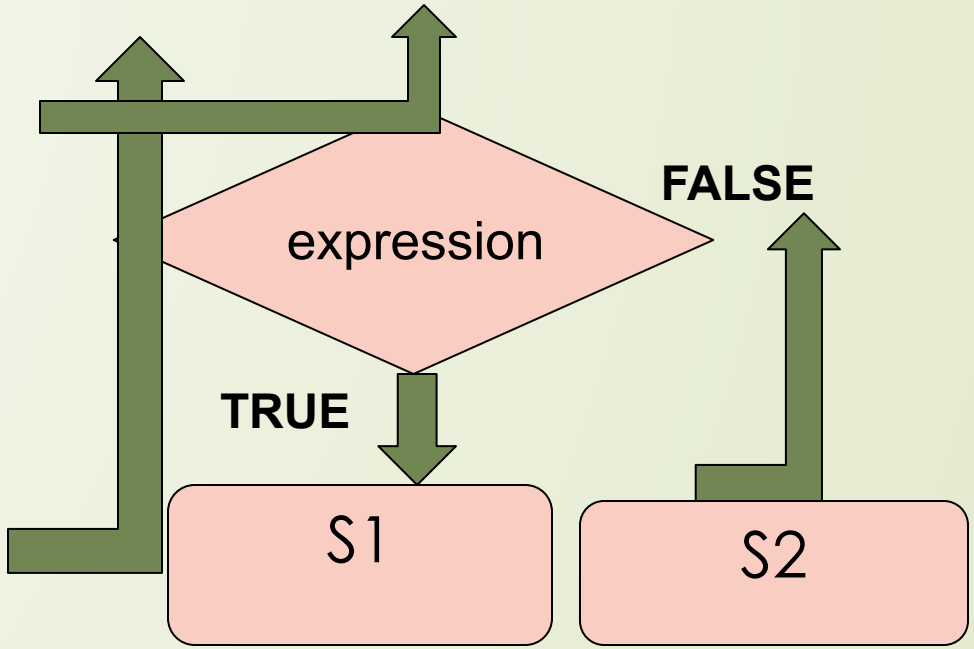



```
n = int(input('n=? '))
i = 1

while (i <= 10) :
    print (n ,'X', i, '=', n*i)
    i = i + 1
print ('done')
```

While Statement


```
while (expression):
    S1
S2
```



1. Evaluate expression
2. If TRUE then
   a) execute statement1
   b) goto step 1.
3. If FALSE then execute statement2.

For Loop

- Print the sum of the reciprocals of the first 100 natural numbers.

```
rsum=0.0# the reciprocal sum

# the for loop
for i in range(1,101):
    rsum = rsum + 1.0/i
print ('sum is', rsum)
```

For loop in Python

- General form

```
for variable in sequence:
    stmt
```

range

- range(s, e, d)
  - generates the list:

    [s, s+d, s+2*d, …, s+k*d]

    where s+k*d < e <= s+(k+1)*d
- range(s, e) is equivalent to range(s, e, 1)
- range(e) is equivalent to range(0, e)

Exercise: What if d is negative? Use python interpreter to find out.

Quiz

- What will be the output of the following program

```
# print all odd numbers < 10
i = 1
while i <= 10:
    if i%2==0: # even
        continue
    print (i, end=' ')
    i = i+1
```

Continue and Update Expr

- Make sure continue does not bypass update-expression for while loops

```
# print all odd numbers < 10
i = 1
while i <= 10:
    if i%2==0: # even
        continue
    print (i, end=' ')
    i = i+1
```

i is not incremented when even number encountered. Infinite loop!!

Strings

- Strings in Python have type str
- They represent sequence of characters
  - Python does not have a type corresponding to character.
- Strings are enclosed in single quotes(') or double quotes(“)
  - Both are equivalent
- Backslash (\) is used to escape quotes and special characters

Strings

```
>>> name='intro to python'
>>> descr='acad\'s first course'
```

- More readable when print is used

```
>>> print descr
acad's first course
```

Length of a String

```
>>> name='intro to python'
>>> empty=''
>>> single='a'
```

\n is a **single** character: the special character representing newline

Concatenate and Repeat

- In Python, + and * operations have special meaning when operating on strings
  - + is used for concatenation of (two) strings
  - * is used to repeat a string, an int number of time
  - Function/Operator Overloading

Concatenate and Repeat

```
>>> details = name + ', ' + descr
>>> details
"intro to python, acad's first course"
```

Indexing

- Strings can be indexed
- First character has index 0

```
>>> name='Acads'
```

Indexing

- Negative indices start counting from the right
- Negatives indices start from -1
- -1 means last, -2 second last, ...

```
>>> name='Acads'
>>> name[-1]
's'
>>> name[-5]
'A'
>>> name[-2]
'd'
```

Indexing

- Using an index that is too large or too small results in "index out of range" error

Slicing

- To obtain a substring
- s[start:end] means substring of s starting at index start and ending at index end-1
- s[0:len(s)] is same as s
- Both start and end are optional
  - If start is omitted, it defaults to 0
  - If end is omitted, it defaults to the length of string
- s[:] is same as s[0:len(s)], that is same as s

Slicing


```
>>> name='Acads'
>>> name[0:3]
```

More Slicing


```
>>> name='Acads'
>>> name[-4:-1]
'cad'
>>> name[-4:]
'cads'
>>> name[-4:4]
'cad'
```

Understanding Indices for slicing

A	c	a	d	s	
0	1	2	3	4	5
-5	-4	-3	-2	-1	

Out of Range Slicing

A	c	a	d	s
0	1	2	3	4
-5	-4	-3	-2	-1

- Out of range indices are ignored for slicing
- when start and end have the same sign, if start >=end, empty slice is returned