MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Computational Thinking and Algorithms

159.171

Python Basics

A revision

Amjed Tahir

a.tahir@massey.ac.nz

Previous contributors: Catherine McCartin & Giovanni Moretti

Python basics

getting a user's input, in string form

```
# Python 3.x
answer = input("prompt goes here ")
```

in form other than a string

```
# Python 3.x
answer = int(input("choose a number "))
answer = float(input("choose a number "))
```

Python basics

`print` is:

an statement in Python 2.x, a function in Python 3.x

e.g.

```
# Python 2.x
print "Z = ", x*7

# Python 3.x
print("Z = ", x*7)
```

Python basics

`print` is:

a statement in Python 2.x, a function in Python 3.x

printing multiple items on a line

```
# Python 2.x form
print "The value in x is ", x
print ("The value in x is " + str(x))

# Python 3.x form
print("The value in x is ", x)
```

Python basics

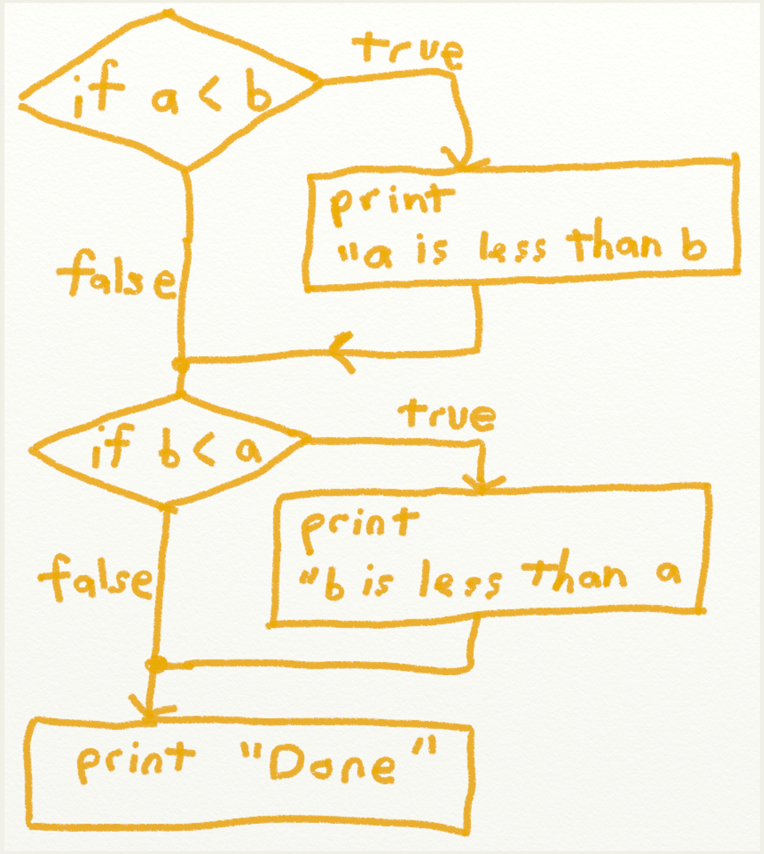
choice:

conditional statements allow us to make choices during program execution

```
a = 4
b = 5
# basic comparisons
if a < b:
  print("a is less than b")
if a > b:
  print("a is greater than b")
```

Python basics

flowchart illustrates logical flow
of program execution



Python basics

```
# if statements:
# less than or equal, greater than or equal
if a <= b:
  print("a is less than or equal to b")
if a >= b:
  print("a is greater than or equal to b")
```

the **<=** and **>=** symbols must used in order, no space between them

=< will not work, nor will **< =**

Python basics

```
# if statements: “equal” and “not equal”

# Equal
if a == b:
   print("a is equal to b")

# Not equal
if a != b:
   print("a and b are not equal")
```

Python basics

don't mix up **=** and **==**

```
# Test whether or not a is equal to 1
# return value is True or False
   a == 1
#  Sets a to the value 1
   a = 1

# this is wrong
a == 1
# this is also wrong
if a = 1:
   print("a is one")
```

Python basics

Indentation matters

```
if a == 1:
    print("If a is one, this will print.")
    print("So will this.")
    print("And this.”)
print("This will always print because it is not indented.")
```

Indentation must be uniform, this code doesn't work.

```
if a == 1:
  print("Indented two spaces.")
    print("Indented four. This will generate an error.")
   print("The computer will want you to make up your mind.")
```

Python basics

using and/or

```
# and (all must be true)
if a < b and a < c:
    print("a is less than b and c")

# or (non-exclusive – any one being true is sufficient)
if a < b or a < c:
    print("a is less than either b or c (or both)")
```

Python basics

Boolean values

`True` or `False`

Boolean valued statements

```
1 + 2 = 3   True
5 < 4      False
```

Boolean valued variables

```
a = True
if a:
   print("a is true")
```

Python basics

```
# How to use the not function
if not(a):
    print("a is false")
```

this is also legal:

```
# How to use the not function
if not a:
    print("a is false")
```

Python basics

```
a = True
b = False

if a and b:
    print("a and b are both true")

if a or b:
    print("at least one of a and b is true")
```

Python basics

```
a = 3
b = 3

# c will be true or false,
# depending if a is equal to b
c = (a == b)

# prints value of c, in this case True
print(c)
```

Python basics

else and **elsif**

```
temp = int(input("What's the temp (C)? "))

if temp > 25:
    print("It is hot outside")
else:
    print("It is not hot outside")

print("Done")
```

Python basics

```
temp = int(input("What's the temp (C)? "))

if temp > 25:
    print("It is hot outside")
elif temp > 20:
    print("It's quite warm")
else:
    print("It is not hot outside")

print("Done")
```

Python basics

What's wrong here?

```
if temperature > 25:
    print("It is hot outside")
elif temperature > 40:
    print("You could fry eggs on the pavement!")
elif temperature < 10:
    print("It is cold outside")
else:
    print("It is ok outside")
print("Done")
```

Python basics

checking text

```
userName = input("What is your name? ")
if userName == "Catherine":
    print("You have a nice name.")
else:
    print("Your name is ok.")

if userName == "Cate" or userName == "Mary":
...
```

Python basics

checking text – case insensitive

```
userName = input("What is your name? ")

if userName.lower() == "cate":
    print("You have a nice name.”)

elif userName.upper() == "MARY":
    print("You have a nice name.")
else:
    print("Your name is OK.")
```

Python basics

repetition:

`for loops`

repeat something a certain number of times

`while loops`

repeat something until something else happens

Python basics

```
for i in range(5):
    print ("I will not miss workshops.")
```

increment variable, can be any legal variable name

```
for i in range(1000):
    print ("I will not miss workshops.")
```

controls how many times the loop is run

Python basics

indentation matters

```
for i in range(5):
   print ("I will not miss workshops.”)
   print (“I will not miss lectures.”)

for i in range(5):
   print ("I will not miss workshops.”)
print (“I will not miss lectures.”)
```

What is the output in each case here?

Python basics

`range` function

`range(5)` returns 5 numbers **`0,1,2,3,4`**

starts at 0 by default

`range(2,8)` returns 6 numbers **`2,3,4,5,6,7`**

starts at 2, ends at 8-1, left index inclusive, right index exclusive

`range(2,12,2)` returns 5 numbers **`2,4,6,8,10`**

starts at 2, ends at 12-2, third parameter is step count

Python basics

counting down

```
for i in range(10, 0, -1):
    print(i)
```

starts at 10, ends at 0+1,

- left index included, right index excluded
- negative step count

prints out 10 numbers

```
10,9,8,7,6,5,4,3,2,1
```

Python basics

nested loops

```
# What does this print?
for i in range(3):
    print ("a")
for j in range(3):
    print ("b")

# What does this print?
for i in range(3):
    print("a")
    for j in range(3):
        print("b")
```

Python basics

keeping a running total

```
total = 0

for i in range(5):
    newNumber = int(input("Enter a number: " ))
    total += newNumber

print ("The total is: " + str(total))
```

Python basics

keeping a running total

```
total = 0
for i in range(5):
    newNumber = int(input("Enter a number:  " ))
    total += newNumber
print ("The total is: " + str(total))
```

IMPORTANT: create & initialise totalling variable **outside** the loop

Python basics

```
# What values of a are printed for these blocks?
a = 0
for i in range(5):
    a = a+1
print(a)

a = 0
for i in range(5):
    a = a + 1
for j in range(5):
    a = a + 1
print(a)
```

Python basics

```
# What is the value of a?
a = 0
for i in range(5):
    a = a + 1
    for j in range(5):
        a = a + 1
print(a)
```

Python basics

```
# print the numbers 0 to 9
for i in range(10):
    print(i)
```

...can be done with a **`while`** loop that looks like this:

```
# using a while loop to print the numbers 0 to 9
i = 0
while i < 10:
    print(i)
    i = i + 1
```

Python basics

```
i = 0
while i < 10:
    print(i)
    i += 1
```

sentinel value

code repeats as long as condition holds

increment i (shorthand version)

what does this while loop do?

```
i = 0
while i < 10:
    print(i)
```

Python basics

```
# Looping until a game is over or user wants to quit
done = False

while not done:
    quit = input("Do you want to quit? ")
    if quit == "y" :
        done = True
    attack = input("Does your elf attack the dragon? ")
    if attack == "y":
        print ("Bad choice, you died.")
        done = True
```

How can we “turn off” second part of loop if user wants to quit?

Python basics

random numbers:

Python uses a module (library) to create random numbers.

You must first **`import`** a module before you use it:

```
import random
```

Python basics

```
import random

# random number from 0 to 49
my_number = random.randrange(50)

# random number from 100 to 200
my_number = random.randrange(100,201)
```

works like **`range()`**, left index inclusive, right index exclusive

Python basics

```
# picking a random item out of a list
my_list = ["rock", "paper", "scissors"]
random_index = random.randrange(3)
print(my_list[random_index])


or


choice = random.choice(my_list)
print(choice)
```