While loops and booleans

CS195 - Lecture 4
Instructor: Dr. V

- while loops
- code blocks, indentation, IndentationError
  - importance of proper indentation in python
  - importance of proper indentation habits in coding in general
  - folding/unfolding code blocks in editor
  - pass
- boolean expressions
- true/false equivalents
- infinite loops
- nested loops

What is the purpose of computers?

- More generally, what is the purpose of technology?

while loop

- what is the purpose of a loop?
    - it's a way to repeatedly do something

```
# as long as [condition] is True, execute [code block]
   [ code before while loop ]

   while [condition] :
      [ code block ]

   [ code after while loop ]
```

while loop

```python

# what do you think this code does?

print('Type a number.')
userResponse = int( input('>> ') )

while userResponse < 100:
    print('No, type a higher number.')
    userResponse = int( input('>> ') )
```

while loop

```python

# what do you think this prints?

x = 1

while x < 5:
    print(f"x = {x};  x * 2 = {x*2}")
    x += 1

print("Done.")
```

while loop

```python

# what do you think this prints?

x = 1

while x < 100:
    print(f"{x = };  {x * 2 = }")     # f-string magic
    x += 1

print("Done.")
```

while loop syntax - code blocks and indentation

```
[ code before while loop ]

while [condition] :
   [ code block ]

[ code after while loop ]
```

- *[code block]* starts after a colon
- every line of code in the same *[code block]* is indented, all in the same exact way
- *[code block]* ends when indentation reverts back

code blocks

```python

# what do you think this prints?

x = 1

while x < 5:
    print("a" * x)
    x += 1

    print("Done.")
```

code blocks

```python

# what do you think this prints?

x = 1

while x < 5:
    print("a" * x)
    x += 1

print("Done.")
```

Code indentation

- in Python:
  - correct indentation is part of the syntax
    - your code will not run without correct indentation
    - or your code might do something you didn't intend
- in other programming languages
  - correct indentation is still VERY important
    - you have to be obsessive about proper indentation
    - otherwise, your code becomes
      - unreadable
      - difficult to debug

while loop - condition

```
[ code before while loop ]

while [condition] :
   [ code block ]

[ code after while loop ]
```

- *[condition]* is evaluated as a boolean expression
  - a boolean expression evaluates to True or False

while loop

```python

# x<5 is a boolean expression - it is either True or False

x = 1

while x < 5:
    print(f"{x = }")
    x += 1

print("Done.")
```

while loop

```python

# what do you think this will do?

x = 1

while True:
    print(f"{x = }")
    x += 1

print("Done.")
```

beware infinite loops

```python

# what do you think this will do?

x = 1

while x > 0:
    print(f"{x = }")
    x += 1

print("Done.")
```

beware infinite loops

```python

# what do you think this will do?

x = 1

while x != 10:
    print(f"{x = }")
    x += 1

print("Done.")





```

beware infinite loops

```python

# what do you think this will do?

x = 1

while x != 10:
    print(f"{x = }")
    x += 2

print("Done.")
```

Boolean conversion

```python

# if your condition is not a boolean expression,
#    it will automatically, get converted to a boolean

while 0:      # equivalent to  if bool(0):
    print("hello")
```

How other types convert to booleans

```python
# what do you think each of these prints?
print( bool(1) )
print( bool(100) )
print( bool(-1) )
print( bool(0) )

print( bool(2.3) )
print( bool(0.0) )

print( bool("hello") )
print( bool("") )

print( bool(None) )


```

How other types convert to booleans

```python

# what do you think this code does?
r = input('>> ')
while r:
    print(f'you said "{r}"')
    r = input('>> ')

print('goodbye')
```

Using the walrus operator

```python

# what do you think this code does?

while r := input('>> '):
    print(f'you said "{r}"')


print('goodbye')
```

How other types convert to booleans

```python

# what do you think this code does?
x = 5
while x:
    print(f'{x = }')
    x -= 1


# what do you think this prints?
print(f'after the loop is over, x is {x}')
```

Using the walrus operator

```python

# what do you think this code does?
x = 5
while x:=x-1:
    print(f'{x = }')



# what do you think this prints?
print(f'after the loop is over, x is {x}')
```

How booleans convert to other types

```
>>> str( True )
'True'
>>> str( False )
'False'
>>> float( True )
1.0
>>> float( False )
0.0
>>> int( True )
1
>>> int( False )
0
>>> 7 + True  # arithmetic operators will auto-cast bool's to int's


```

Boolean operators

```python
x, y = 13, 10

# what do you think each of these prints?
print( x > y )
print( x < y )
print( x >= y )
print( x <= y )
print( x <= x )
print( x == y )
print( x != y )

# what about these?
print( 10 < x < 20 )
print( 10 <= x < 20 )

```

Boolean operators

```python
x, y = "abc", "def"

# what do you think each of these prints?
print( x == y )
print( x != y )

print( x > y )
print( x < y )
print( x >= y )
print( x <= y )
print( x <= x )
```

pass

- you can have an empty code block
  - use the keyword pass; e.g.:

```
[ code before while loop ]
while [condition] :
   pass
[ code after while loop ]
```

- usually pass is used as a placeholder
- sometimes it is used because no code is necessary in the code block

empty code block using pass

```python

# what do you think this code does?

PASSWORD_PROMPT = 'Please enter a strong password: '

while (password:=input(PASSWORD_PROMPT)).isalnum():
    pass

print(f"{password=}")
```

Assignment 4

- create an echo bot that does the following:
  1. it says "Hello."
  2. records user input as *userResponse*
  3. as long as user input isn't "q" or "Q"
     - it says "well *{userResponse}* right back at ya"
     - records user input as *userResponse*, again
     - repeat step 3