

• Branching & Iteration

Today

- string object type
- branching & conditionals
- indentation
- iteration and loops

STRINGS

- letters, special characters, spaces, digits
- enclose in **quotation marks or single quotes**

```
1 hi = "hello there"
```

- **concatenate strings**

```
1 name = "andy"
2
3 greet = hi + name
4
5 greeting = hi + " " + name
```

- **do some operations on a string**

```
1 silly = hi + " " + name * 3
```

INPUT/OUTPUT: print

- used to **output** stuff to console
- keyword is **print**

```
1 x = 1
2 print(x)
3 x_str = str(x)
4 print("my fav num is", x, ".", "x =", x)
5 print("my fav num is " + x_str + ". " + x_str)
```

INPUT/OUTPUT: `input("")`

- prints whatever is in the quotes
- user types in something and hits enter
- binds that value to a variable

```
1 text = input("What is your name?")
2
3 print(5 * text)
```

- **input gives you a string** so must cast if working with numbers

```
1 num = int(input("Type a number... "))
2
3 print(5 * num)
```



COMPARISON OPERATORS ON:

int, float, string

- i and j are variable names
- comparisons below evaluate to a Boolean

```
1 i > j
2 i >= j
3 i < j
4 i <= j
5 i == j # equality test, True if i is the same as j
6 i != j # inequality test, True if i is not the same as j
```

LOGIC OPERATORS ON bools

a and b are variable names (with Boolean values)

not a → **True if a is False**

False if a is True

a and b → **True if both are True**

a or b → **True if either or both are True**

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

CONTROL FLOW - BRANCHING

```
if (condition):  
    # expression  
    # expression
```

```
if (condition):  
    # expression  
    # expression
```

```
else:  
    # expression  
    # expression
```

```
if (condition):  
    # expression  
    # expression
```

```
elif (condition):  
    # expression  
    # expression  
  
else:  
    # expression  
    # expression
```

(condition) has a value True **or** False

evaluate expressions in that block if (condition) **is** True

INDENTATION

- matters in Python
- how you denote blocks of code

```
1 x = float(input("Enter a number for x: "))
2 y = float(input("Enter a number for y: "))
3
4 if x == y:
5     print("x and y are equal")
6
7     if y != 0:
8         print("therefore, x / y is", x/y)
9
10 elif x < y:
11     print("x is smaller")
12
13 else:
14     print("y is smaller")
15
16 print("thanks!")
```



keeps going right, takes you
back to this same screen,
stuck in a loop

Legend of Zelda - Lost Woods

```
1 if exit right:  
2     # set background to woods_background  
3     if exit right:  
4         # set background to woods_background  
5         if exit right:  
6             #set background to woods_background  
7             # and so on and on and on...  
8         else:  
9             # set background to exit_background  
10        else:  
11            # set background to exit_background  
12    else:  
13        # set background to exit_background
```



Legend of Zelda - Lost Woods

keeps going right, takes you
back to this same screen,
stuck in a loop

```
1 while exit_right:  
2     # set background to woods_background  
3 else:  
4     # set background to exit_background
```

Control Flow: while LOOPS

```
while condition:  
    # expression  
    # expression
```

- **condition** evaluates to a Boolean
- if **condition** is True, do all the steps inside the while code block
- check **condition** again
- repeat until **condition** is False

while LOOP EXAMPLE

You are in the Lost Forest.



Go left or right?

```
1 n = input("You are in the Lost Forest\n*****\n*****\n :)\n*****\n*****\nGo left or right? ")  
2  
3 while n == "right" or n == "Right":  
4     n = input("You are in the Lost Forest\n*****\n***** ***\n (舅 °□°) ω ~ L \n*****\n*****\nGo left or right? ")  
5 print("\nYou got out of the Lost Forest!\n)o/")
```

CONTROL FLOW: while and for LOOPS

- iterate through numbers in a sequence

```
# more complicated with while loop
n = 0

while n < 5:
    print(n)
    n = n + 1

# shortcut with for loop
for n in range(5):
    print(n)
```

CONTROL FLOW: for LOOPS

```
for your_variable in range(some_number):  
    # expression  
    # expression
```

- each time through the loop, **your_variable** takes a value
- first time, **your_variable** starts at the smallest value
- next time, **your_variable** gets the previous value + 1
- etc.

range (**start**, **stop**, **step**)

- default values are **start = 0** and **step = 1** (if omitted)
- loop until value is **stop - 1**

```
1 mysum = 0
2
3 for i in range(7, 10):
4     mysum += i
5
6 print(mysum)
7
8
9 mysum = 0
10
11 for i in range(5, 11, 2):
12     mysum += i
13
14 print(mysum)
```

break STATEMENT

- immediately exits whatever loop it is in
- skips remaining expressions in code block
- exits only innermost loop!

```
1 while condition_1:  
2     while condition_2:  
3         # expression_A  
4         break  
5         # expression_B  
6         # expression_C
```

break STATEMENT

what happens in this program?

```
1 mysum = 0
2
3 for i in range(5, 11, 2):
4     mysum += i
5
6     if mysum == 5:
7         break
8     mysum += 1
9
10 print(mysum)
```

for VS while LOOPS

for loops

- know number of iterations
- can end early via break
- uses a counter
- can rewrite a for loop using a while loop

while loops

- unbounded number of iterations
- can end early via break
- can use a counter but must initialize before loop and increment it inside loop
- may not be able to rewrite a while loop using a for loop