# Python Objects
# Conditionals
# Midterm Review

# Announcements

- No class on Tuesday
- Extra office hours Monday and Wednesday – check the calendar
- Midterm 1 is next week. To prepare, view:
  https://ucsb-cs8-f18.github.io/exam/e01/
- Midterm format
  - Concept questions - fill in the blanks, multiple choice and short answers
  - Generalize a given function
  - Implement a new function and test using the pytest framework
  - Turtle graphics – understand how to pass parameters to functions
  - Trace through code and give the output
  - Variable and functions involving data types covered so far:
    int, float, bool str, list, tuple
  - Python Modules (using modules and writing your own)
  - Python objects vs classes (today)
  - Conditionals –if-else (today)

# Python Objects

- Every piece of data in Python is an object
- Think of an object as a generic container to store data on a computer's memory
- Every object has a type and value
- e.g. $x = 3$ creates an object of type int and value 3

# Python Classes

- A class is a formal description of a type
- It describes all operators and methods that can be applied to objects of the class
- It provides a standard way to create new objects of that class (via its constructor)

```
pets = list()
pets.append("cat")
pets = pets + ["goat", "goldfish"]
pets = pets *2
pets.count("goat")
pets.sort()
```

- You can define your own data type using classes (just like Turtle or Fraction) - that's object oriented programming but we won't go into it for now

# Relational operators

- Remember: = is the Python assignment operator
  - It is a command to evaluate the right-hand side and make the variable on the left refer to that result
  - In math (not Python!), = is a claim that two expressions are equal
- == is the Python operator that tests for equality
  - Other relational operators: > >= < <= != (the last one means "not equal")
  - They return bool (Boolean) values

# Concept Test

What is the output of the following code?

```
a = 3
b = (a != 3)
print(b)
```

- A. `True`
- B. `False`
- C. 3
- D. Syntax error

# Functions returning Boolean values

For each of the following write a function that takes one parameter `x`, and returns True if the following condition is satisfied, otherwise returns False

- A. `x` is an integer
- B. `x` is negative (assuming its an integer)

How about writing a function that returns True if x is a negative integer false otherwise?

# Logical operators

- The logical operators take one (`not`) or two (`and`, `or`) bools and return a bool
- An expression involving `not` produces `True` if the original value is `False`, and `False` if the original value is `True`
- `And` produces `True` exactly when both of its operands are `True`
- `or` produces `True` exactly when at least one of its operands is `True`

# Concept Test

I would like an expression that evaluates to `True` exactly when at least one of the following two conditions is true: (1) `a` and `b` are equal, (2) when `a` has value 5. Which of these expressions does that?

- A. `a == b == 5`
- B. `a == b or a == 5`
- C. `a == b and a == 5`
- D. `a == (b == 5)`

# Python Operators

## Precedence

( ) Highest

**

-

* / %

+ -

> < ==

not

and

or

= Lowest

## Caution Level

set equal to =

divide /

remainder %

power **

is equal to ==

as usual: * + > < - ( )

It's not worth remembering all these %+/* things!

I'd go with parentheses over precedence

# Concept Test

What is the value of the expression at the bottom of the code? (Remember that `not` has the highest precedence, then `and`, then `or`.)

```
a = True
b = False
c = True
not a and b or c
```

- A. `True`
- B. `False`

# More functions returning Boolean

For each of the following write a function that takes one parameter `x`, and returns true if the following condition is True, otherwise returns false

- A. `x` is an integer and its value is negative
- B. `x` is an odd integer (don't make assumptions about the value of `x`)

How would you modify the above code so that the function additionally prints a message when `x` is odd (instead of returning true)?

# If and If Else

```
if <condition>:
   <sequence of statements>
```

If the condition evaluates to True, execute sequence of statements, otherwise jump to end of if block

```
if <condition>:
   <sequence of statements-1>
else:
   <sequence-of-statements2>
```

If the condition evaluates to True, execute code inside if block, otherwise execute code in the else block

# Concept Test

```
x = 5
if x > 2:
    x = -3
    x = 1
else:
    x = 3
    x = 2
```

- A. $-3$
- B. 1
- C. 2
- D. 3
- E. 5

# Midterm Review

Which of the following is NOT true about "variables" in Python

- A. A variable is a name that refers to a value
- B. Variables let us store ("remember") values so we can use them in several places
- C. The value of a variable once assigned cannot be changed
- D. Either of the following statements can be used to store the value 3 in variable x: `x=3 OR 3=x`
- `E. Options C and D`

# Assignment statement

- The assignment statement lets us give a value to a variable
- Form: `variable = expression`
- Two steps:
  1. Evaluate the expression on the right-hand side to get a result
  2. Make the variable on the left-hand side refer to that result

Trace through the following code and write the value of x and y in each case

```
x = 3
y = (x==3)
x = y+1
point = (x, y)
name = "Suzie"
lst = [x, y, point, name]
```

# Concept Test

What is the value of y after the execution of this code?

```
x = "cat"
y = x*2
x = "mouse"
```

- A. cat
- B. catcat
- C. mouse
- D. mousemouse
- E. The statement results in an error

# Print vs Return

```
def sayHello():
    print("Hello!!")

x = sayHello()
```

Identify all the function definitions

Identify all the function calls

What is the value of x when the above code is executed?

- A. "Hello!!"
- B. None
- C. Code results in an error

# Print vs. Return

```
def sayHello():
    return "Hello!!"

x = sayHello()
```

What is the value of x when the above code is executed?

- A. "Hello!!"
- B. None
- C. Code results in an error

# Identify all the function calls

```
1   import turtle
2
3   def square(t,side):
4       regularShape(t,side,4)
5
6   def pentagon(t,side):
7       regularShape(t,side,5)
8
9   def regularShape(t,side,n):
10      for i in range(n):
11          t.forward(side)
12          t.right(360/n)
13
14  if __name__ == "__main__":
15      t = turtle.Turtle()
16      t.speed(0)
17      for s in range(10,20,1):
18       square(t,s)
```

# Good luck with the midterm!