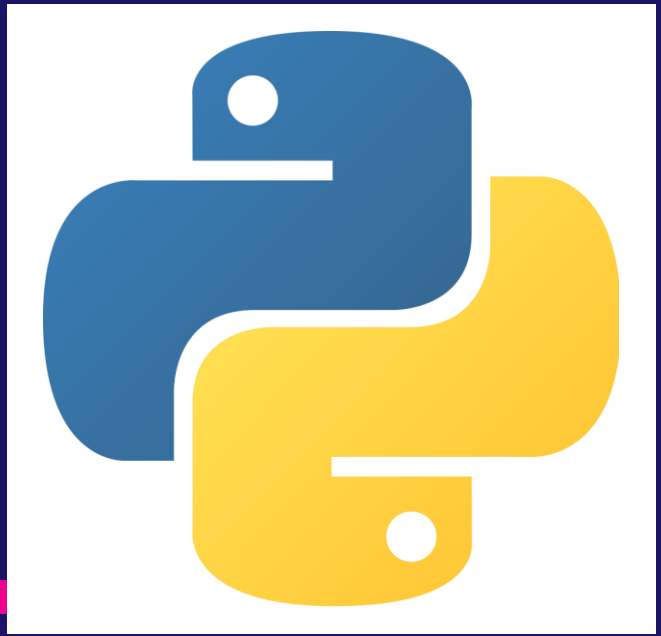
Python I

Welcome to Hack The RAM!



Welcome to Hack The RAM!

repl.it

- Go to repl.it and create an account
- This website will allow you to create and run programs online
- Collaboration feature allows your team to collaborate on a program

Program file

- The white half of the screen labeled “main.py” is your program file
- This is where you will type all of your code

Console

- The console is the black half of the screen
- This is where any information that you print will appear after you run the program

Function Notation

- A function is a piece of code that performs a task for you.

```
function1()
function2(param)
function3(param1,param2)
```

Print Statements

- The easiest way to output anything or communicate with the user is to use a print statement
- In python, we do this through the use of the **print()** function

```
print(“Hello world!”)
```

Data Types

- **string** - a line of symbols, usually used to express words
  - Ex: “hello” ‘a’ “$ % & #” “54321”
  - Can contain symbols, numbers, spaces, letters
  - Cannot not do math with strings Ex: “5”/”3” = ERROR
- **int** - a whole number
- **float** - a decimal
- **boolean** - True or False (always capitalized)

YOU CANNOT COMBINE DIFFERENT TYPES: “4” + 3 = ERROR

Combining Strings - Concatenation

- To combine strings, you can use a comma
- `print('Grand' + "ma")` = `"Grandma"`

- This is known as string concatenation

`print("Happy" , "Birthday")` → Happy Birthday

Operators

- Addition (+)

  `print(5 + 2)` → 7
- Subtraction (-)

  `print(5 - 2)` → 3
- Multiplication (*)

  `print(5 * 2)` → 10
- Division (/)

  `print(5 / 2)` → 2.5

Variables

- Variables are names that store values
  - Can store any data type (string, int, etc), other variables, and more
- Giving a variable a value is known as **assignment**

```
num = 4
```

Getting input

- To get data from the user, use the **input()** command
- You can give the user a message about what they should input by putting text in between the parentheses
- You can store what the user inputs in a variable
  - Input is always stored as a string

```
name = input("Please enter your name: ")
print(name)
```

Comments

- Any text followed by a # will create a comment
- Comments are for you to understand the code, and will not have any effect on the program.

```
print(“Hello World) #this outputs hello world
```

Converting variable types

- You cannot add strings and integers together
- To convert from string to int, use the **int()**
  - int(“3”) → 3
- To convert from int to string, use the **str()**
  - str(3) → “3”

```
print(str(1) + “1”) → 11
print(1 + int(“1”)) → 2
num = int(input(“Enter a number: ”))
```

“\n”

“\n” is used to put space between two lines of output. This is useful for spacing out the output so it looks cleaner and more readable to the user.

```
print("Hello Mark")
question = input("How is your day going: ")
```

Output:

```
Hello Mark
How is your day going?
```

```
print(“Hello Mark\n”)
Question = input(“How is your day going?”)
```

Output:

```
Hello Mark

How is your day going?
```

Practice Program

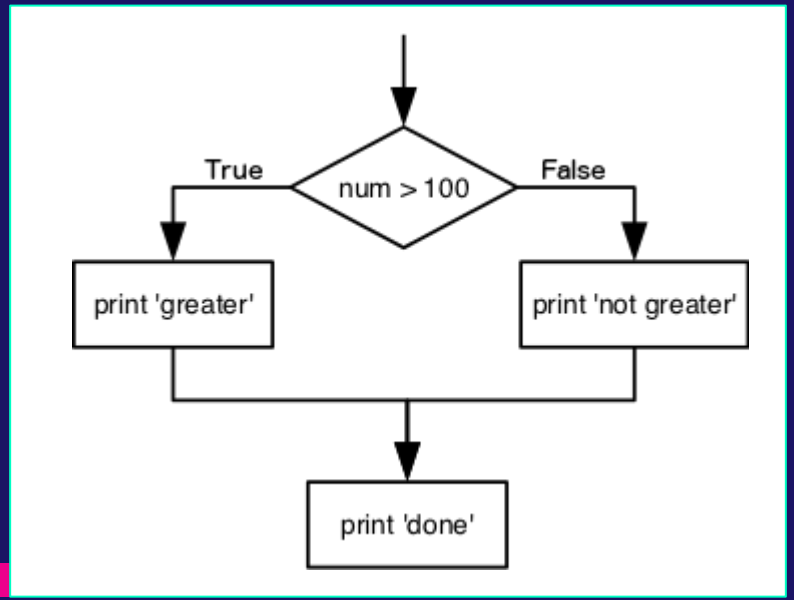
Create a program which takes input in Celsius and converts it to Fahrenheit

- Use this formula to convert
  - Fahrenheit = 1.8*Celsius + 32

SAMPLE OUTPUT:

```
Enter the temperature (Celcius): 22
Temperature in Fahrenheit: 71.6
```

Conditionals



Boolean & Relational Operators

- Used to compare the values of two numbers or variables
  - Use **==** to check if two values are equivalent
  - Use **>**, **<**, **>=**, **<=** to compare values
  - Use **!=** to check if two values are ***not*** equal

```
x = 3
y = 4
print(x >= y) → False
```

Logical Operators

- There are 3 logical operators: **or**, **and**, **not**
- OR - requires one of the two statements to be true for the overall statement to be true

  `print(True or False)` → **True**
- AND - requires both statements to be true for the overall statement to be true

  `print(True and False)` → **False**

if statements

- 2 parts: a **condition** (in parentheses) and a **body** (indented)
  - The code in the body only runs only if the condition is TRUE
- After the condition, you need a colon(:)

```
name = “Mantavya”

if(name==”Mantavya”):

print(“It’s true”)
```

- The print statement would execute only if the variable `bool` in the parentheses is True. If `bool` is False, the block of code is skipped

Python II

If/Else Statements

- An else statement will only run if the condition is FALSE
- The `else` should match the indentation of the `if` and should also be followed by a colon

```
if(9+9 == 17):
    print(“That’s true.”)
else:
    print(“That’s false.”)
```

- The above statement would output “`It is false.`”

If/Elif/Else

- **Elif** (short for else if) operates similar to a second if statement
  - Executes only if first if statement is not met, so ORDER MATTERS
- An if statement can have only one else block, but it can have many elif blocks

```
num = 29
if(num > 100):
    print(“Too big”)
elif(num > 50):
    print(“Just right”)
else:
    print(“Too small”)
```

- What would output if num = 70? If num = 100? If num = 50?

Nested If Statements

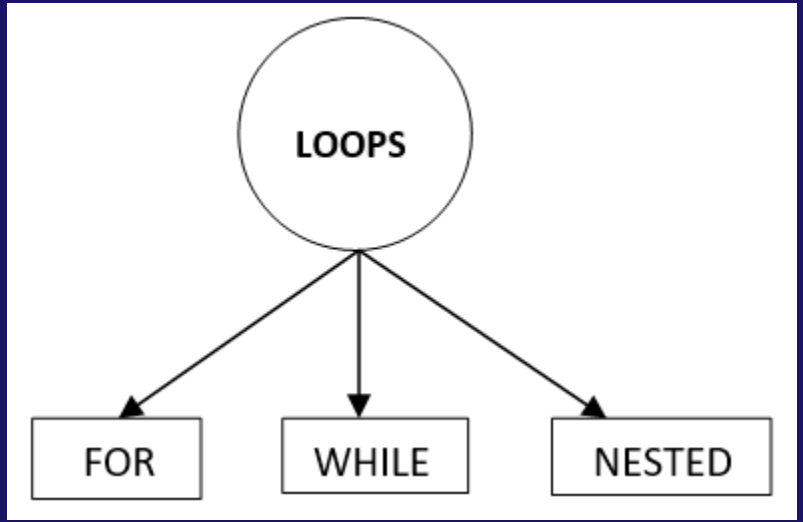
- Putting an if statement inside another if statement is known as a nested if statement

```
if(condition1):
    if(condition2):
        print(“Hi”)
```

- The code would output `Hi` only if condition1 and condition2 were true
- Alternatively , you could use a statement with **and** to do the same

```
if(condition1 and condition2):
```

</>Loops and Lists



Advanced Operators

- The increment operator is commonly used in loops
- The operator += will increment a variable
  - `x += 2` is equivalent to `x = x + 2`

    ```
    num = 3
    num+=1
    print(num) → 4
    ```
- You can also use **-=, *=,** and **/=**

Lists

- A list is a type of variable that can store multiple values
  - A list can store ints, strings, and even other lists

Ex..

```
fruits = []
fruits = [“apple”, “orange”, “grape”]
```

Accessing Values in a list

- Use the [] to access a value from a list
- print(fruits[0]) → apple

- List start numbering at 0

- You can also go backwards if easier
- print(fruits[-1]) → grape

```
fruits = [“apple”, “orange”, “grape”]
```

List functions

```
fruits = [“apple”, “orange”, “grape”]
```

myList = any list

- len(myList) - returns the number of items in myList
- myList.append(val) - adds val to the end of the list
- min(myList) - returns the minimum value in a list
- max(myList) - returns the maximum value in a list
- sort(myList) - sorts the values in the list (lowest to highest)
- insert(index, val) - inserts val into myList at the specified index

Intro to Loops

- Loops will run the same block of code multiple times
- Two basic types of loops: **while** loops and **for** loops

While loops

- While loops work in a similar way to an if statement
- The body will continue to run and repeat itself until the condition is false

```
num = 0
choice = “n”
while choice == “n”:
        choice = input(“Enter n to keep incrementing: “)
        num+=1
```

While loops cont..

```
money = 10
while money>0:
        money -=1
        if money<3:
                print(“You’re running out of
money”)
```

Output:

You’re running out of money
You’re running out of money

For-Each loops

- You can use a For loop to loop through each element in a list (using `in`)

```
fruitBowl = [“apple”, “orange”, “grape”]
for fruit in fruitBowl:
    print(fruit) →  apple
                    orange
                    grape
```

- `fruit` (in this case) represents each element in the list during one run of the code in the loop, so the loop will print out each item in the list. You can rename it to whatever seems appropriate.

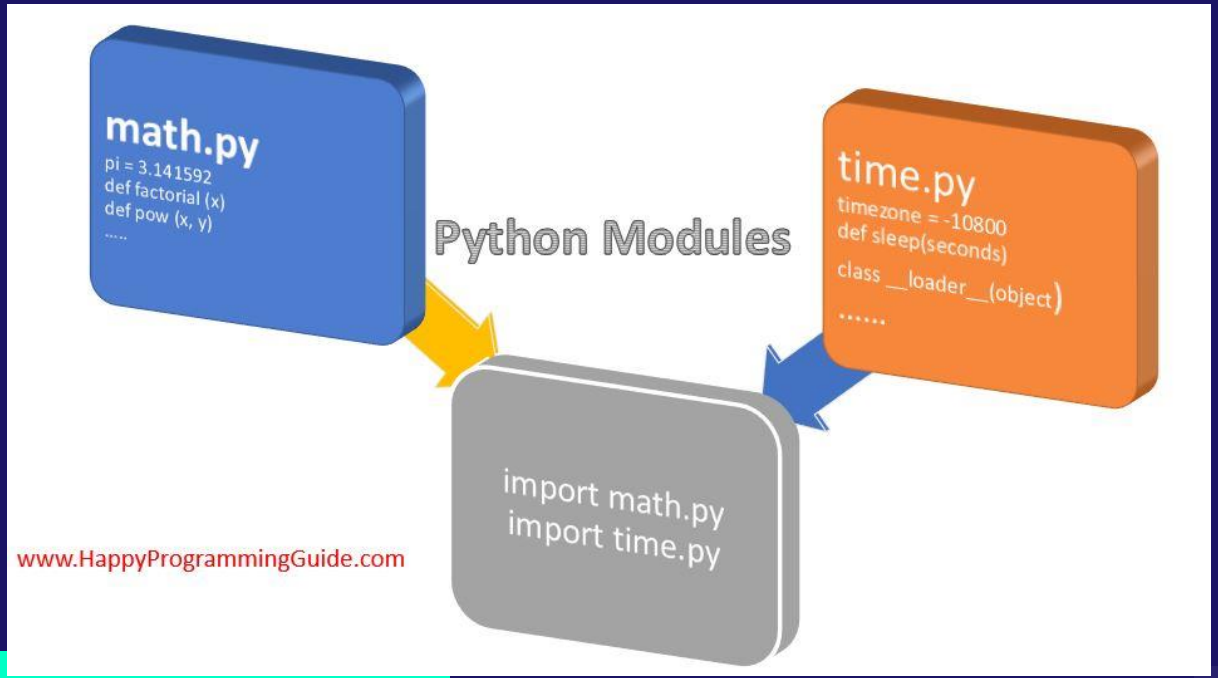
Writing Functions

- Functions are used to simplify code and make it reusable.
- Functions can take in values and return values

```
number = add(4,3)
print(number)

def add(num1+num2) :
        return num1+num2
```

Importing Modules



Importing Modules

- Python does not have more advanced mathematics built into it
- To access them, you can import libraries or **modules** of additional information
  - Two of the most common are **math** and **random**
- Importing is usually done at the top of the program and looks like this:

```
import math, random
```

The Random Module

- The random module allows your program to generate random numbers
  - Random number - **random.randint(min, max)**
  - Random choice from list - **random.choice(myList)**

Project 0: Guess the Number

Write a program where the computer generates a random number between 1-50 and the user will try and guess what the number is. The program should output whether their guess is correct, too high, or too low.

```
Guess a number between 1 and 50: 17
    Too low. Try again.
Guess a number between 17 and 50: 36
    Too high. Try again.
Guess a number between 17 and 36: 24
    Correct! Great job!
```

Bonus step: Give the user a limited number of guesses

Project 1: Rock, Paper, Scissors

Write a program where the user plays versus a computer. The computer will randomly generate either Rock, Papers, or Scissors, and the program will ask the user which option they will pick.

```
Rock, Paper, Scissors, SHOOT!
        What do you pick? scissors

Cpu: Paper                  You: Scissors
- - - - - You Win! - - - - - -
```

Bonus Step: Allow the user to play again when finished.