↑↑↓↓←→←→ BA start sheet

Var: value/string stored in a container

String: text

Integer: whole num

float: decimal num

Modual; use line to access, lets yo use more functions

boolean: true/false

tuple: list that cannot be changed

Function: collection of code grouped together

while loop: loop if condition is met

Python

0 - 38.21

To print

print("words") = words

print("line\nline") = line
line

print("line\"") = line"

escape

Print(var_name + "_sucks") = Var sucks

combine

vars

var_name = "Var"

functions

print(var_name.lower()) = var
.upper() = VAR

[can combine]

.isupper() = false (not all upper)
print(len(var_name)) = 3 (characters)
print(var_name[0]) = V [pick character]
.index("V") = 0 (where it is, can use words)
.replace("v", "b") = bar

start end

numbers
can do math + PEMDAS

%10 = mod, gets remainder

print(str(5) + "is good")
↕ convert to string

abs: absolute value

pow(4,6) = 4^6

max(4, 6) = 6 (higher)
min(4,6) = 4 (lower)
round(3.2) = 3

access more functions ↙

from math import *

floor(3.7) = 3 (lowest, round down)
ceil(3.7) = 4 (round up)
sqrt(36) = 6 (square root)

input() ← get use input
Put prompt in

can put var container
int(num) ← converts to integer
float(num) ← converts to float
friends = ["kevin", "kara", "jim"] ← list (can do multi data types)
str(num) ← converts to string

lists

Print(list[0]) = list value 1
(can do negative)

[1:] = value 1 and onwards
[1:3] = up to but not including 3

friendo[1] = changed value

list1.extend(list2) ← adds 2 lists
list1.append(add me) ← ends to end
list1.insert(1, add) ← what
where
list1.remove("x")
list1.clear
list.pop ← removes last
list.index ← gives where person is in list
list.count ← how many in list
list.sort ← alphabatize / ascending
list.reverse() ← flip
list2 = list1.copy ← copy list

If statements

```
is_male = true

if is_male:
    print("you are male")
else:
    print("you are not male")
```

```
is_male =
is_tall =

if is_male or is_tall:
    print("you are tall, male or both")
else
    print("you aren't tall or male")
```

or one or more must be true

and all must be true

```
elif is_male and not(is_tall)
    print("you are a short male")
```

elif = else if

not inverts

comparing if

```
def max_num(num1, num2, num3)
    if num1 > num2 and num1 > num3:
        return num1
    elif num2 > num1 and num2 > num3:
        return num2
    else:
        return num3

print(max_num(3, 4, 5))
```

comparisons
==
!=
>
<
>=
<=

Calculator test

```
num1 = float(input("number?"))
op = (input("enter op"))
num2 = float(input("number2?"))

if op == "+":
    print(num1 + num2)
elif...
```

Dictionary

```
month = {
    "Jan": "January",
    "Feb": "February",
}
print(month["Jan"])
```

default value

```
.get("Jan")
.get("Jan", "no")
```

--- while loop ---

```
i = 1
while i <= 10:
    print(i)
    i += 1
print("done with loop")
```

add on to i

2:32:43

For loop — can be array — var changes each loop

```
for letter in "word":
    print(letter)
```

w = o r d

```
for index in range(10):
    print(index)
```

= 1 2 3 ... or 3,10 — can also be var list, or array

len(array) ← how many in an array

range(len(array)) — individualizes

```
print(friends[index])
```

Exponent Function

```
print(2**3)
```

2³

loop as many times as this

```
def raise(base, power):
    result = 1
    for index in range(power):
        result = result * base
    return result
print(raise(x,x))
```

2d lists (lists in lists)

```
Big_list = [
  [1, 2, 3]
  [4, 5, 6]
  [7, 8, 9]
  [0]
]
```

column

row

```
print(Big_list[0][0])
```

0 = 1

```
for row in Big_list:
    for col in row:
        print(col)
```

for loop for

Translator

```
def translate(phrase)
    translation = ""
    for letter in phrase:
        if letter in "AEIOUaeiou":
            translation = translation + g
        else:
            Translation = translation + letter
    return translation
print(translate(input("phrase?")))
```

comments /// 3.04.17

words

Try except

```
try:
    code goes here
except:
    code for if it's wrong
```

To catch errors

```
ZeroDivisionError:
ValueError:
```

Make into a var

as err

Reading files

```
open("file.name", "r")
```

can and should put in var

- "r" - read
- "w" - write
- "a" - append
- "r+" - read and write

```
varname.close()
print(varname.readable)
```

print(varname.read())

.readline() ← read line one after another

.readlines() ← makes an array

.readlines()[1] ← which line

writing/appending to files

"a"

varname.write("added")

\n for new line

"w" ← overwrites entire file

can make new file

can put html in file

3:28:24

Modules

import filename ← for homemade

print(filename.fun())

list of python modules (versions)
[official]

Also can just look it up

▽ External libraries
  ▽ (version)
    ▽ Lib

Install modules

pip - package manager

☆ in CMD
  pip install [name]

where modules go

▽ Lib
  ▽ site packages

mod.

classes / objects

class

in file (student.py)

init func

class student:

defining

def __init__(self):

self.nam = name (self, name, major)

in file (app.py) ← object

from student import student

file class

student1 = student("Jim", "Business", 3.1)

print(student1.name)

can put func in classes

inheritance

class New(old):

(also import before

Python interpreter

cmd / terminal

python3