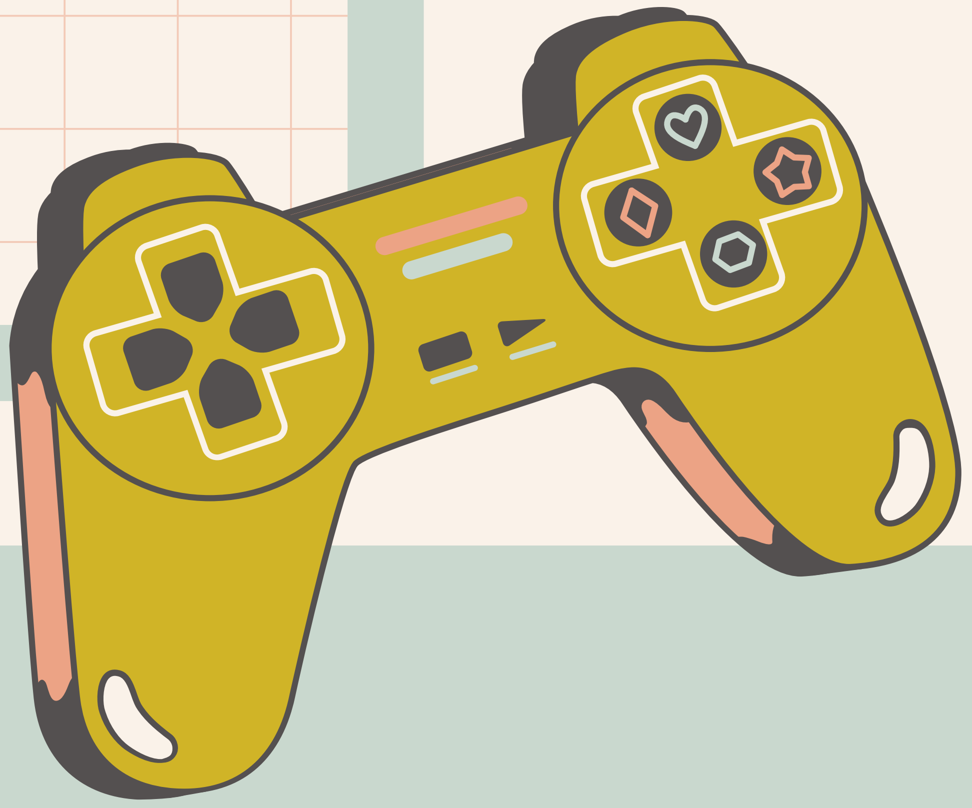
LOOPING AND HANDLING ERROR

Data Science Track - Whitespace



OUR TEAM :

Andhika Utomo

Dean Setyawan

Heru Stiawan

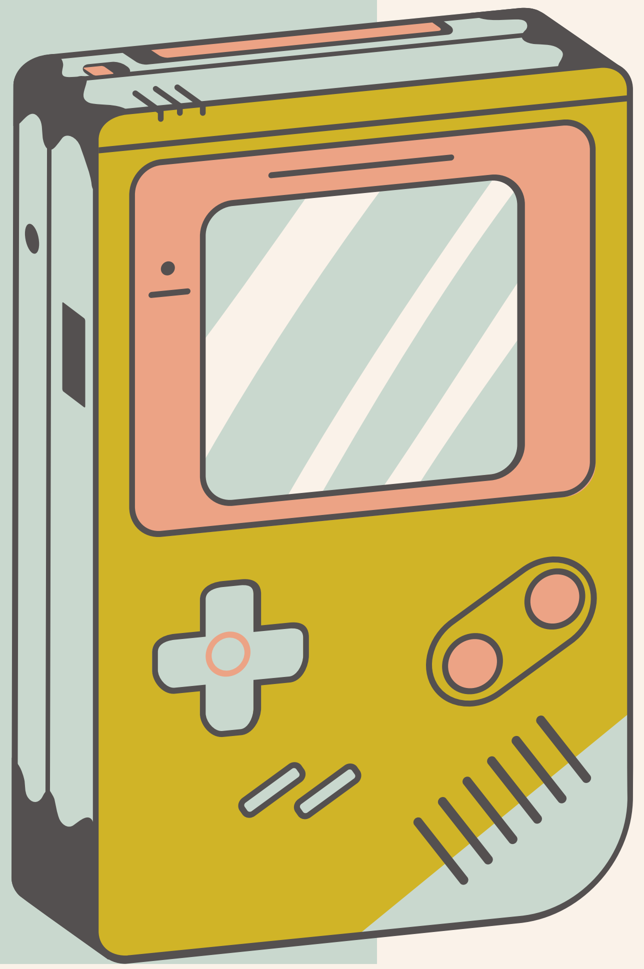
M Haikal Febrian p

Raisya Amanah N

FOR FUNCTION IN PHYTON

For is a fung that can loop each type of variable in the form of a collection or sequence.

- The variable in question can be a list, string, or range.
- if a list or sequence contains experession, it will be evaluated first.
- then the first item in the order / list will be diassigned as a variable iterating_var afterwards, the state data block will be executed, continuing to the next item, repeating until the sequence.



EXAMPLE:

```
visual_design=
['Photoshop','Ilustrator','corel draw']

for perangkat in visual_design:
    print("Software Editing
{}".format(perangkat))
```

OUTPUT:

```
Software Editing Photoshop
Software Editing Ilustrator Software
Editing corel draw
```

```
for i in range(5):
  print(i)
```

OUTPUT:

```
0 1 2 3 4
(in vertical posision)
```

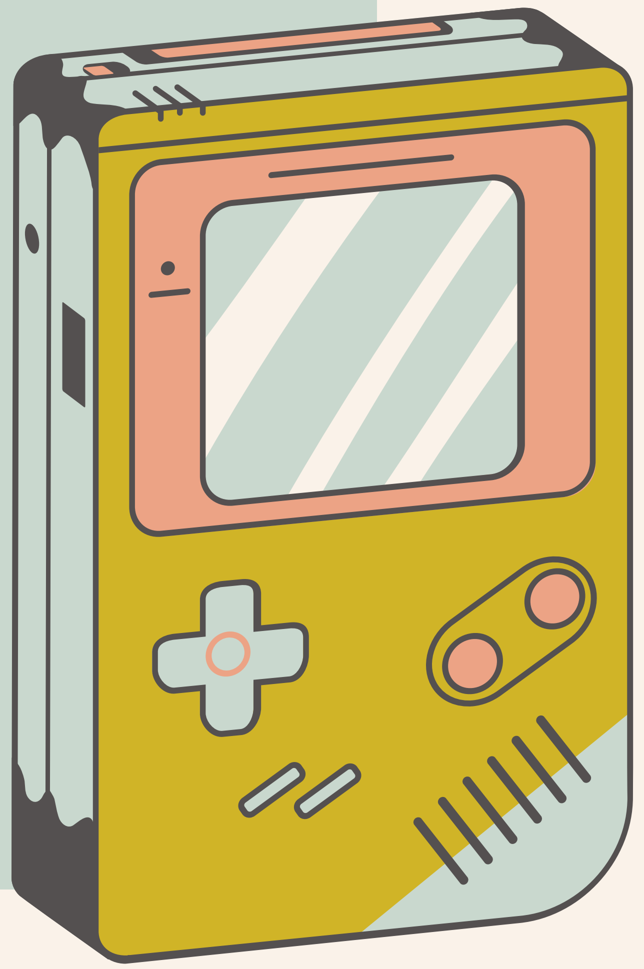
```
for i in range(1,10,2):
  print(i)
```

OUTPUT:

```
1 3 5 7 9
(in vertical posision)
```

FOR LOOP NESTED FUNCTION IN PHYTON

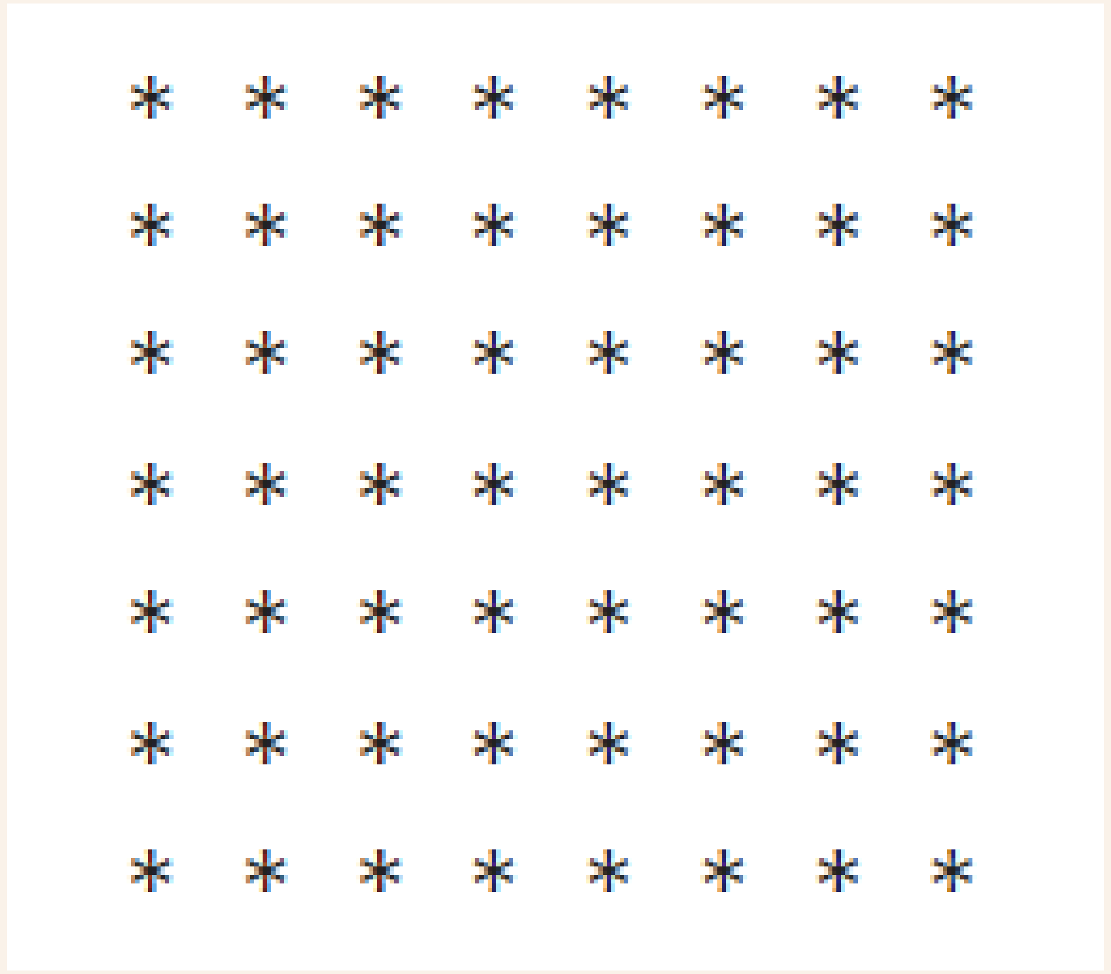
A nested loop has one loop inside of another. These are typically used for working with two dimensions such as printing stars in rows and columns as shown below. When a loop is nested inside another loop, the inner loop runs many times inside the outer loop.



FUNCTION FOR IF ELSE EXAMPLE:

```
for row in range(10):
    for col in range(10):
        if row == 0 and col <= 7:
            print("*", end = " ")
        elif row == 1 and col <= 7:
            print("*", end = " ")
        elif row == 2 and col <= 7:
            print("*", end = " ")
        elif row == 3 and col <= 7:
            print("*", end = " ")
        elif row == 4 and col <= 7:
            print("*", end = " ")
        elif row == 5 and col <= 7:
            print("*", end = " ")
        elif row == 6 and col <= 7:
            print("*", end = " ")
    print()
```

OUTPUT:

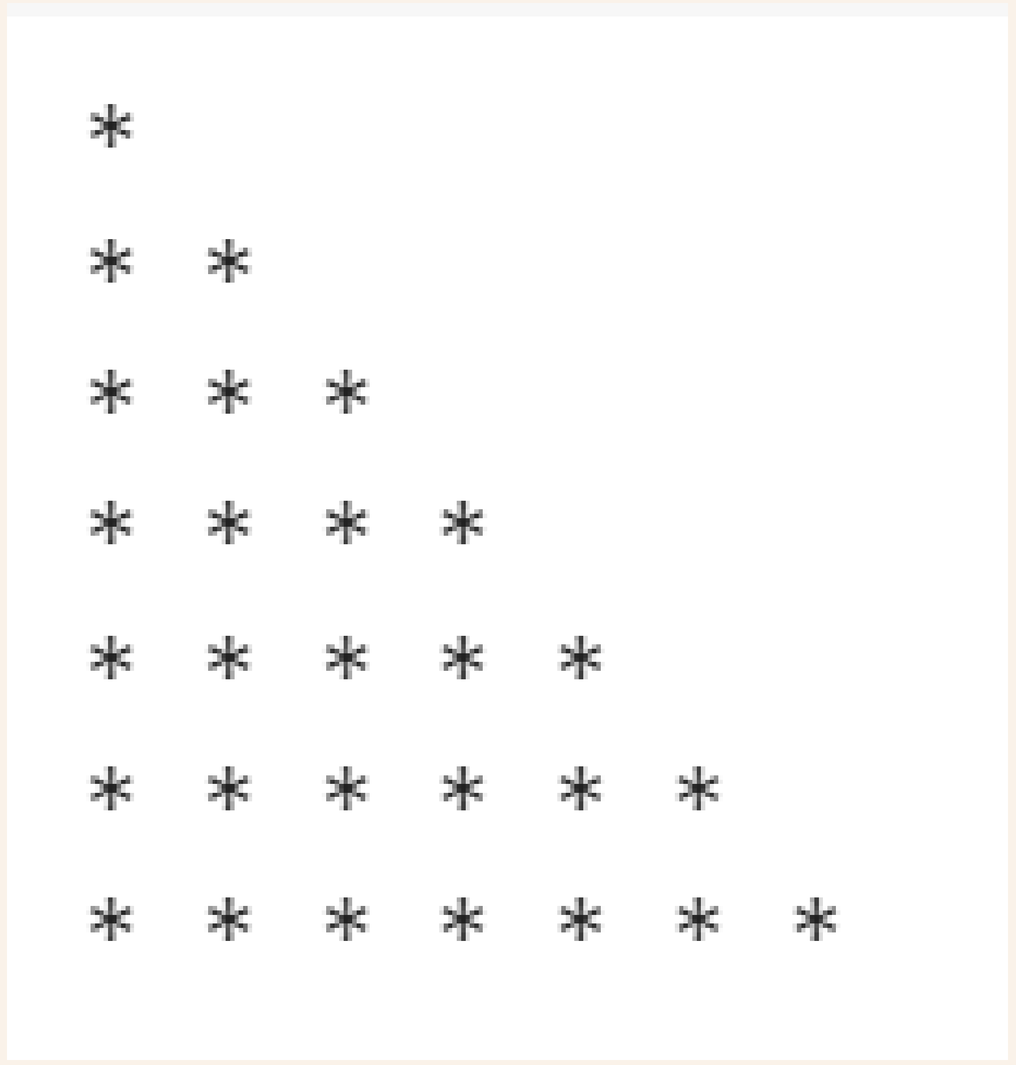


```
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
```

FUNCTION FOR IF ELSE EXAMPLE:

```python
for row in range(10):
    for col in range(10):
        if row == 0 and col <= 0:
            print("*", end = " ")
        elif row == 1 and col <= 1:
            print("*", end = " ")
        elif row == 2 and col <= 2:
            print("*", end = " ")
        elif row == 3 and col <= 3:
            print("*", end = " ")
        elif row == 4 and col <= 4:
            print("*", end = " ")
        elif row == 5 and col <= 5:
            print("*", end = " ")
        elif row == 6 and col <= 6:
            print("*", end = " ")
```

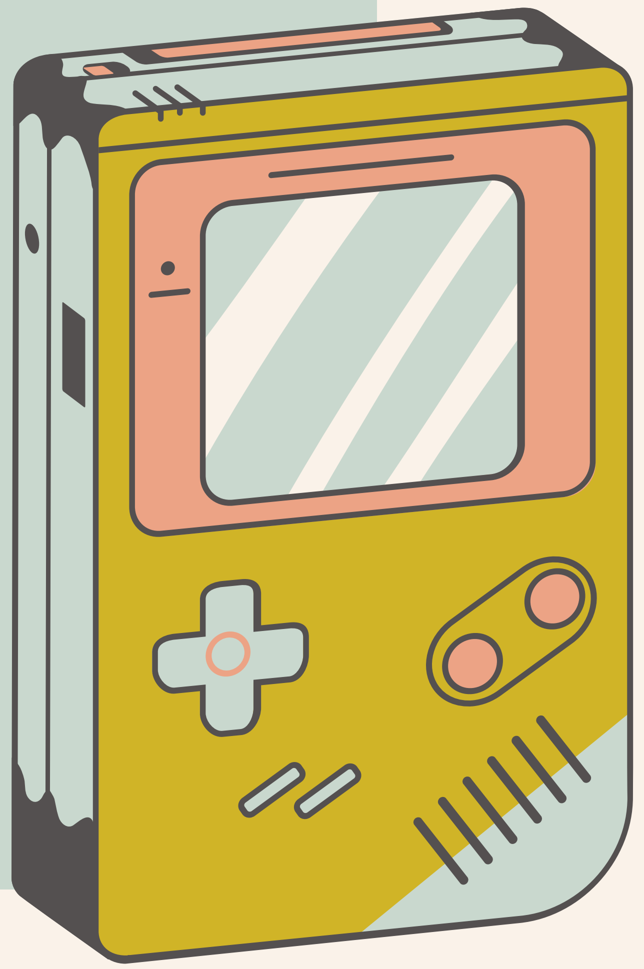
OUTPUT:



```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

WHILE FUNCTION IN PHYTON

- While in the python language is used to execute statements as long as the given conditions are met (True)
- conditions can be any expression, and please keep in mind that true in phytons includes all values non_zero
- when the condition becomes false, the program will proceed to the line after the statement block



FUNCTION WHILE EXAMPLE:

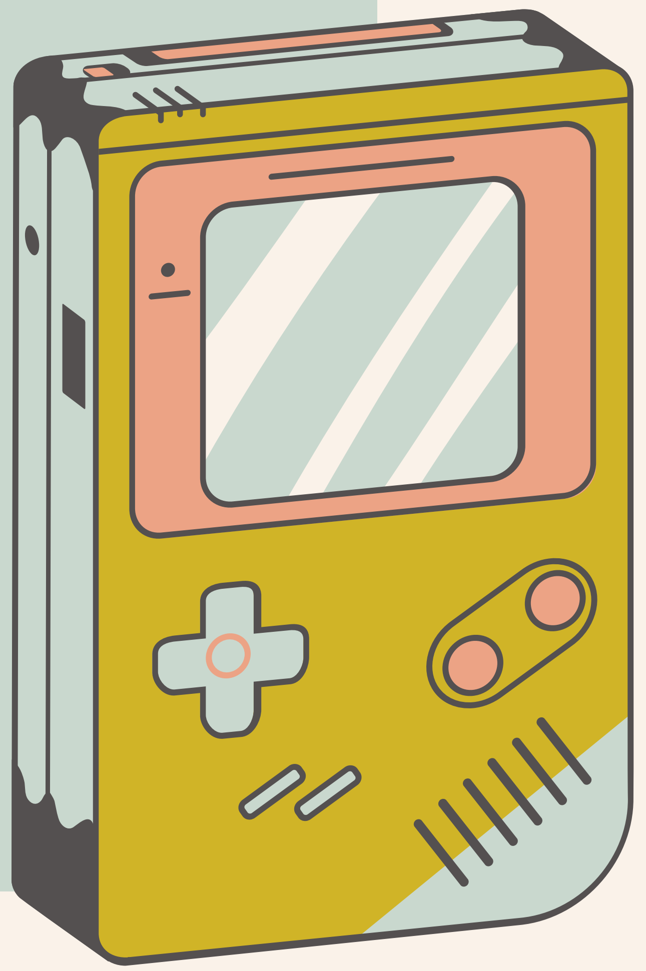
```
angka= int(input())
while (angka < 100):
    print("bilangan terbaik
{}".format(angka))
    angka = angka + 5
```

OUTPUT:

```
85
bilangan terbaik 85
bilangan terbaik 90
bilangan terbaik 95
```

BREAK FUNCTION IN PHYTON

- the break statement stops the loop then exits, followed by executing the statement after the looping block.
- if you have a tiered loop, the break will stop the loop by the level or where it is located.
- but if it is placed in a loop with a second depth, for example, only that loop will stop, not with the main loop.



FUNCTION BREAK EXAMPLE:

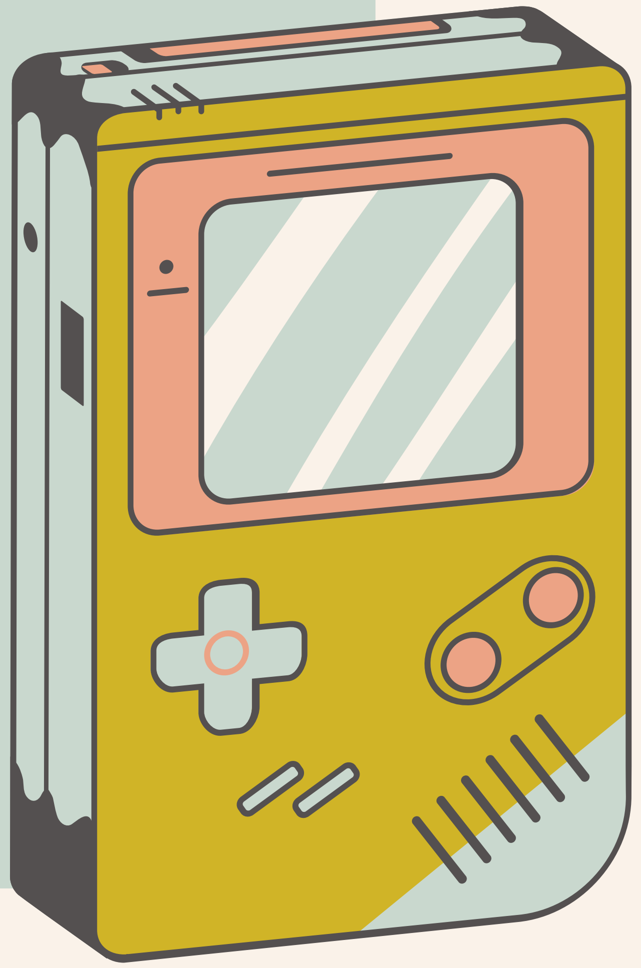
```
x = "Ramadhan Jazz Festival 2022"
for i in x:
    if i == x[23]:
        break
    print(i,end="")
```

OUTPUT:

```
Ramadhan Jazz Festival
```

CONTINUE FUNCTION IN PHYTON

- The continue statement stops the current iteration
- then continues it to the next iteration
- statements that lie between continue to the end of the loop block



FUNCTION CONTINUE EXAMPLE:

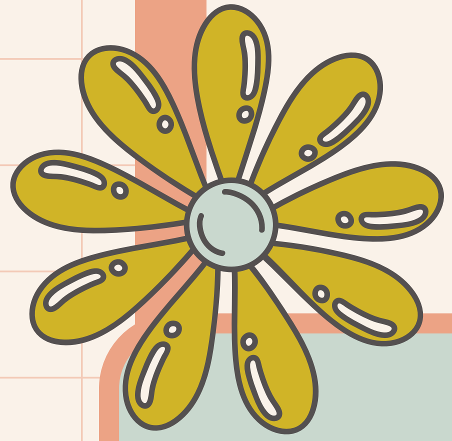
```
x = "Ramadhan Jazz Festival 2022"
for i in x:
    if i == x[23]:
        continue
    print(i,end="")
```

OUTPUT:

```
Ramadhan Jazz Festival 0
```

FOR LOOP

for loop is a loop that is used if the number of iterations or the number of iterations is known



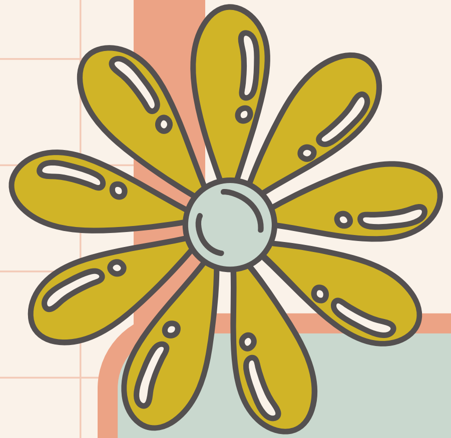
EXAMPLE

example for loop to display only odd number :

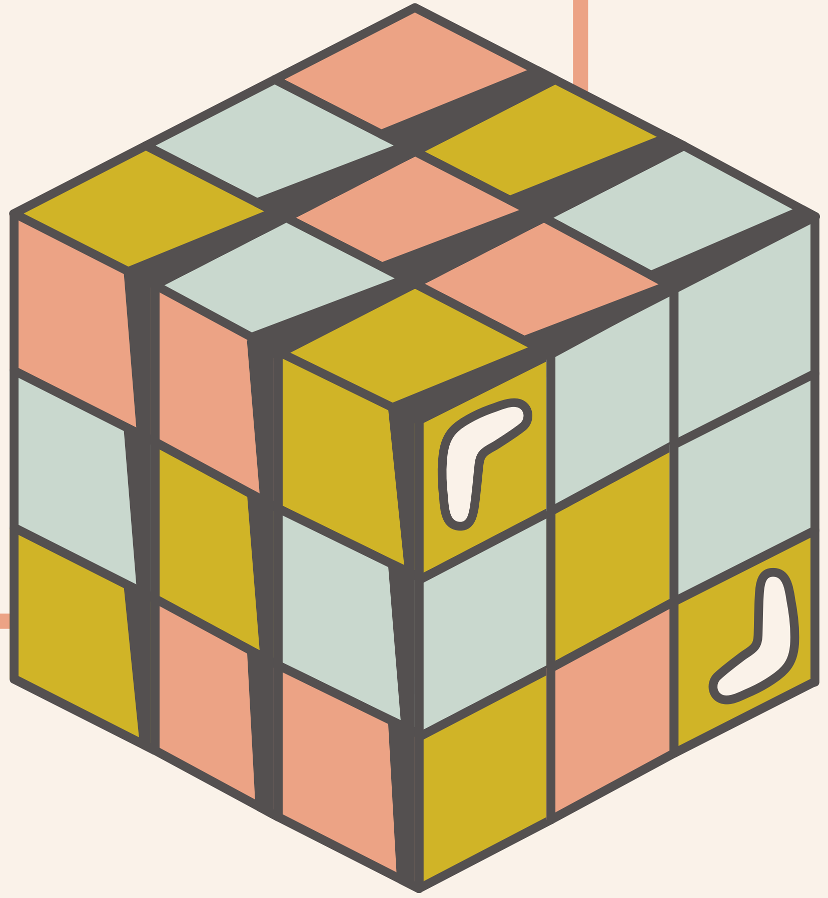
```
for i in range(1,9):
  if i % 2 == 1:
    print("{} adalah Bilangan Ganjil".format(i))
```

WHILE LOOP

while loop is is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. so, while loop will continue looping as long as condition is True

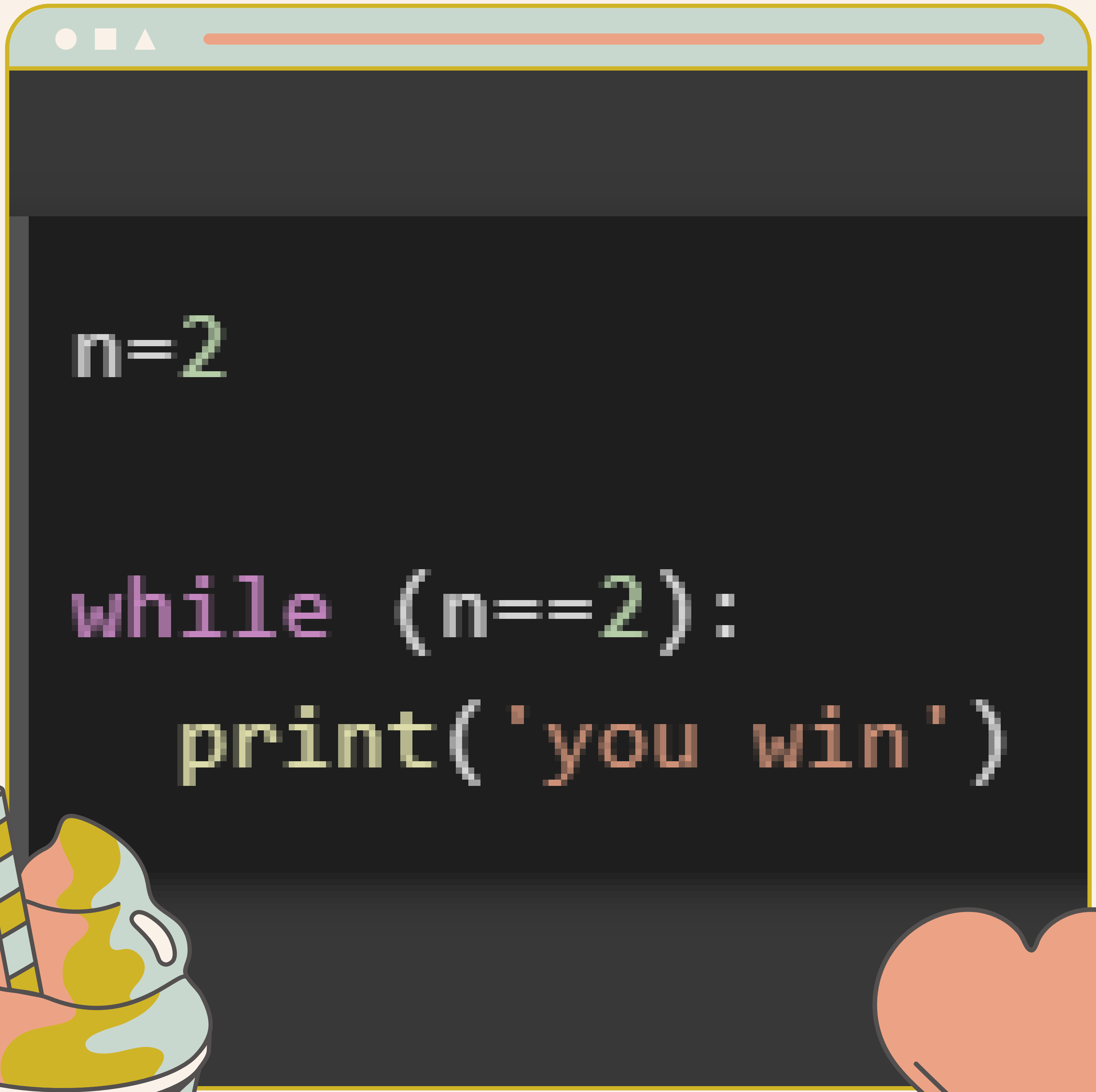


WHAT IS THE MEANING OF THAT?



EXAMPLE

next to this text is a very simple while loop. because n=2 fulfill the condition of while loop so you win text will be print endless till we stop its run time using stop button.

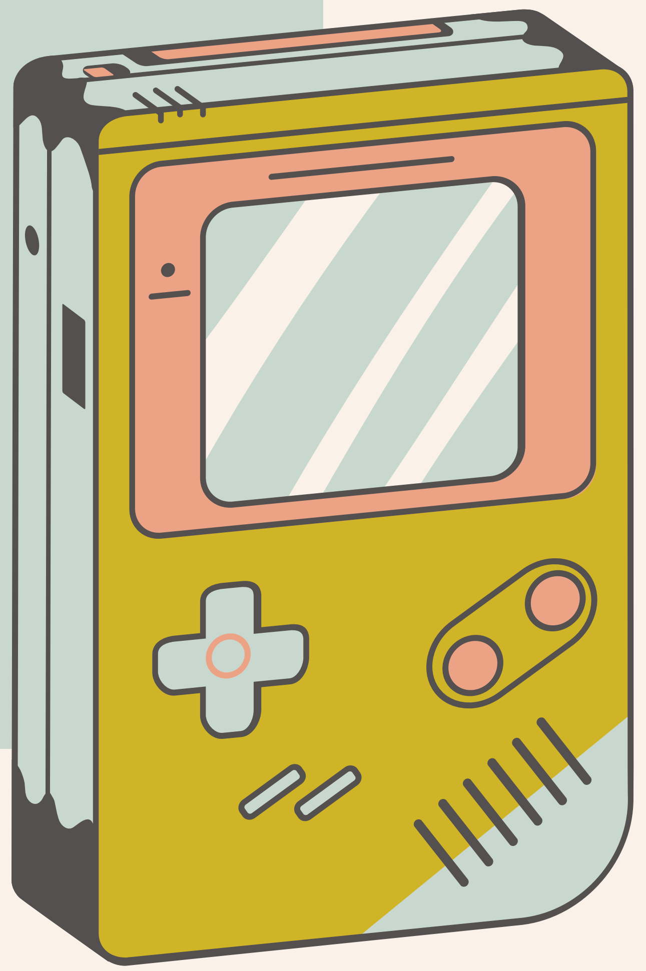


```
n=2

while (n==2):
    print('you win')
```

HOW TO FIX IT?

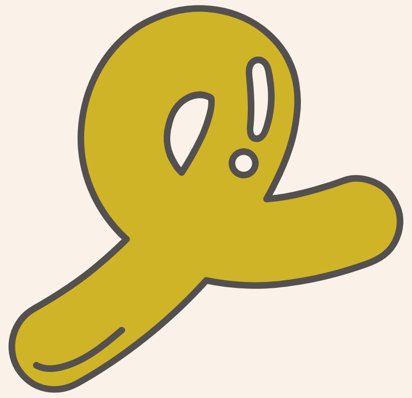
FIXING INFINITE LOOP IN WHILE LOOP

- you can gives function (addition, substraction,multiplication,ect) to change variable we will use for while condition so in the future its condition can be False or unfulfil condition so loop stop
- you can gives stop condition



WHILE ELSE

The while syntax can be added with the else syntax which is used to execute program code when the expression test evaluates to false.



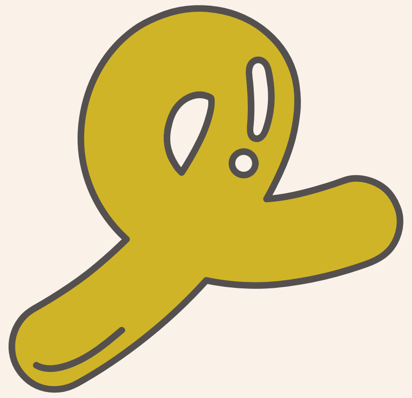
WRITING WHILE ELSE

systematic writing while syntax with else :

```
while loop_expression:
     command_expression
else:
     else_command
```

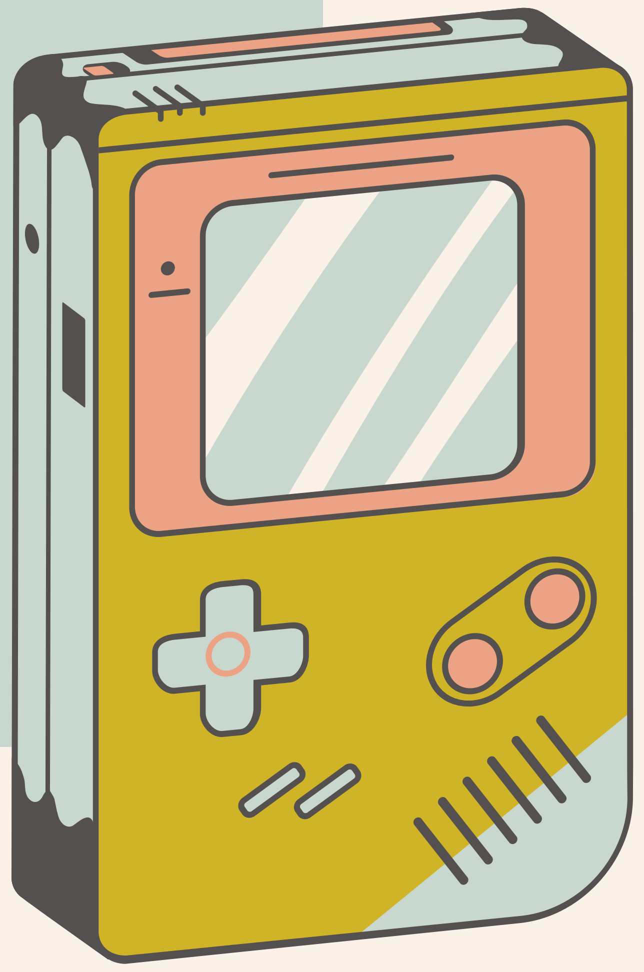
Example :

```
angka = 5
while (angka < 15):
  angka = angka + 2
  print(angka)
else:
  print("Selesai")
```

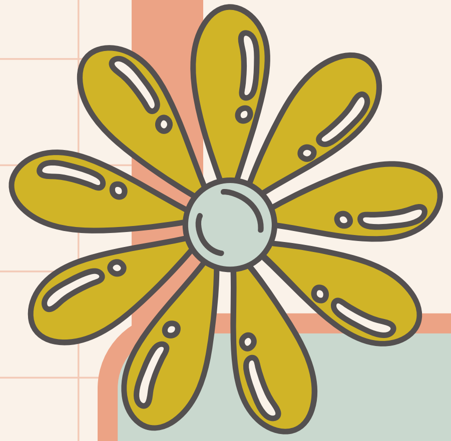


DIFFERENT BETWEEN WHILE AND FOR LOOP

- for loop will continue looping till reach its limit we already spesific
- while loop will continue looping as long as its value true
- so, we can say that for loop already spesific when will end in its function syntax so dont need to add function to stop it and while will continue looping without end if data still fulfill its condition so need to add another function to stop while loop



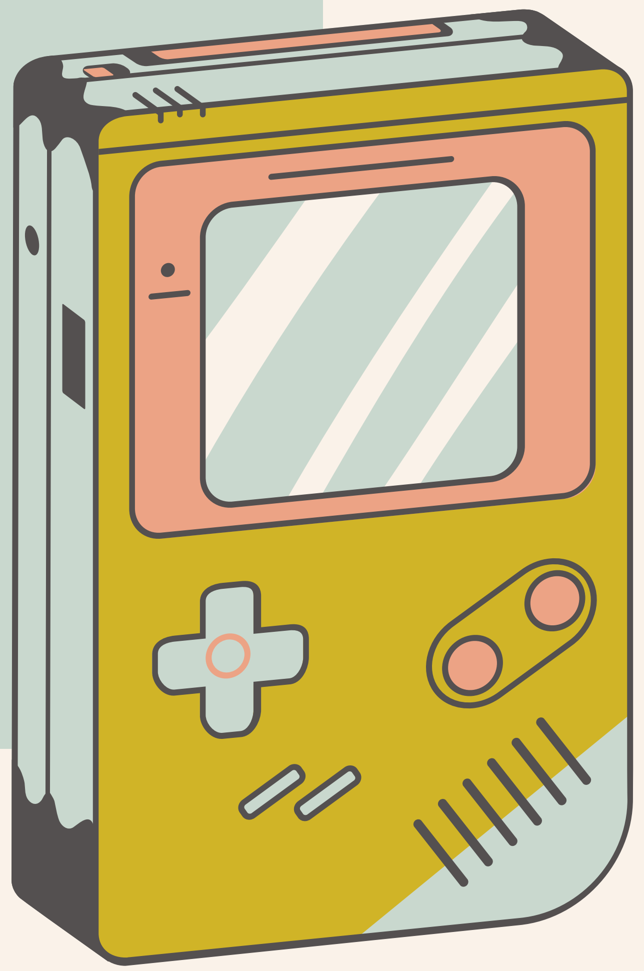
pass is a statement in python that does not have any assignment. Does not instruct the system to do a single thing. It exists, but it's as if it doesn't exist



PASS

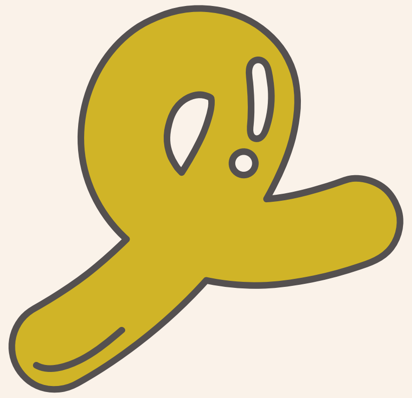
IMPORTANT POINT FROM PASS

- Pass is a null operation, so nothing happens when called
- In simple terms this pass function is if we want a statement but do nothing, just continue to the next statement
- You could say this pass is used to construct a loop



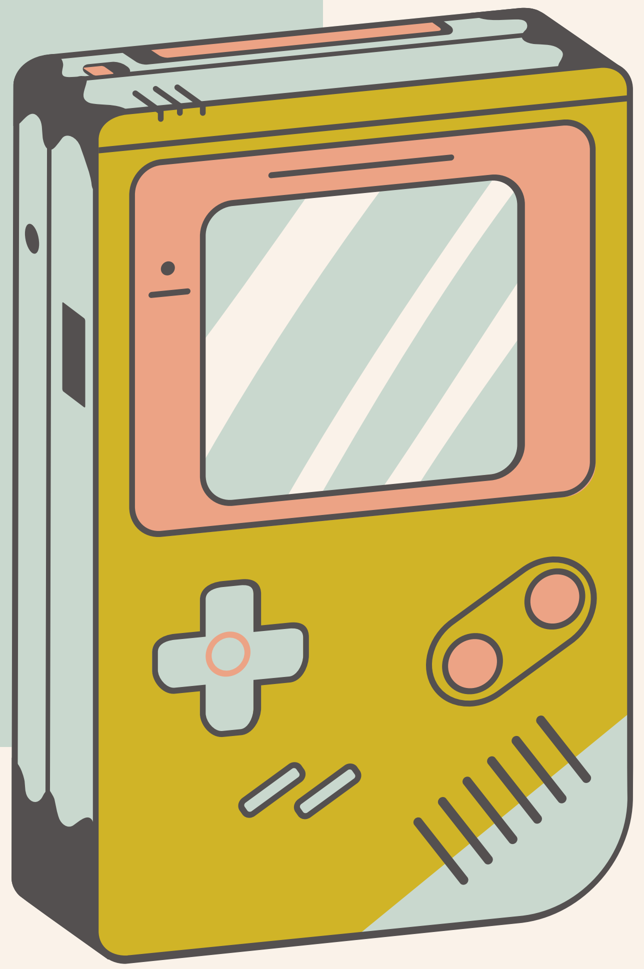
EXAMPLE OF PASS

```
import sys
n=""

while(n!="exit"):
  try:
     n = input("Masukkan angka ")
     print("Dapat angka
{}".format(int(n)))
  except:
    if n=="exit":
      pass
    else:
      print("sudah error
{}".format(sys.exc_info()))
```



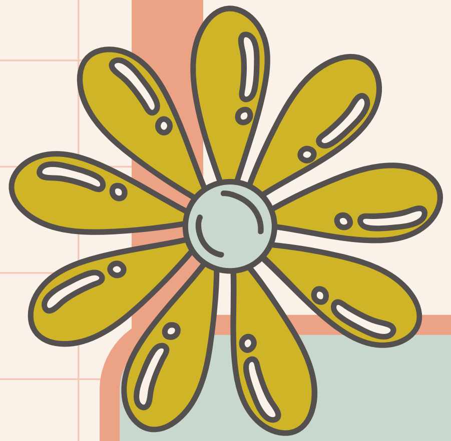
EXPLANATION

so from the program code it can be concluded that only int is displayed otherwise there will be an error, the program will continue to run until we input the word 'exit' and the program will run even though there is an error in it

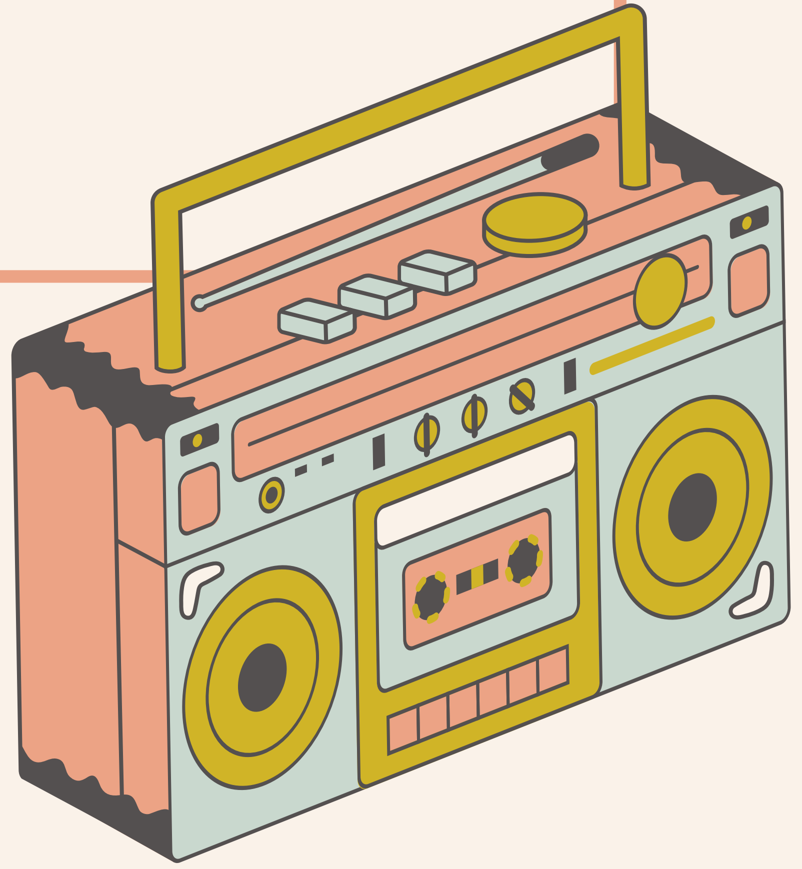


LIST COMPREHENSION

List comprehensions are used to create lists with inline loops and if. as well as an easy way to define and create lists in Python



How to Make List Comprehension



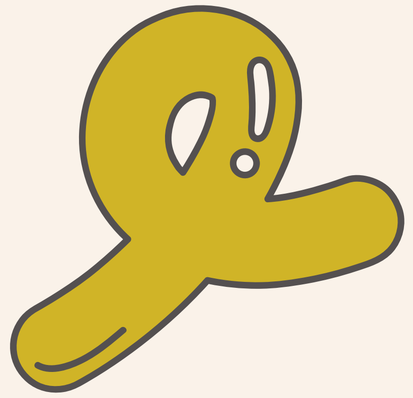
List comprehensions consist of an expression followed by a for statement enclosed in brackets [ ]. Using list comprehensions we can create lists automatically in one command line. This is very useful if the list members we want to create is quite a lot.

EXAMPLE OF LIST COMPREHENSION

```
number = [2,3,4,5,6]

root = []

for i in number:
  root.append(i**(1/2))
print(root)
```

explanation:

so from the code, it will create a list containing the results of the root of the number in the number

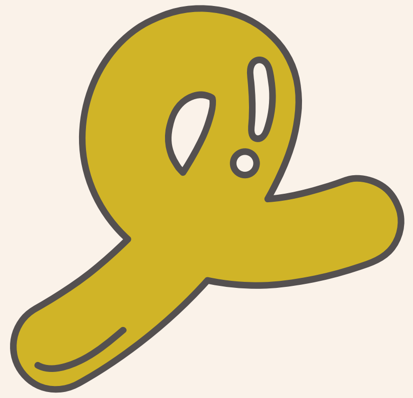


ANOTHER WAY WRITING OF LIST COMPREHENSION

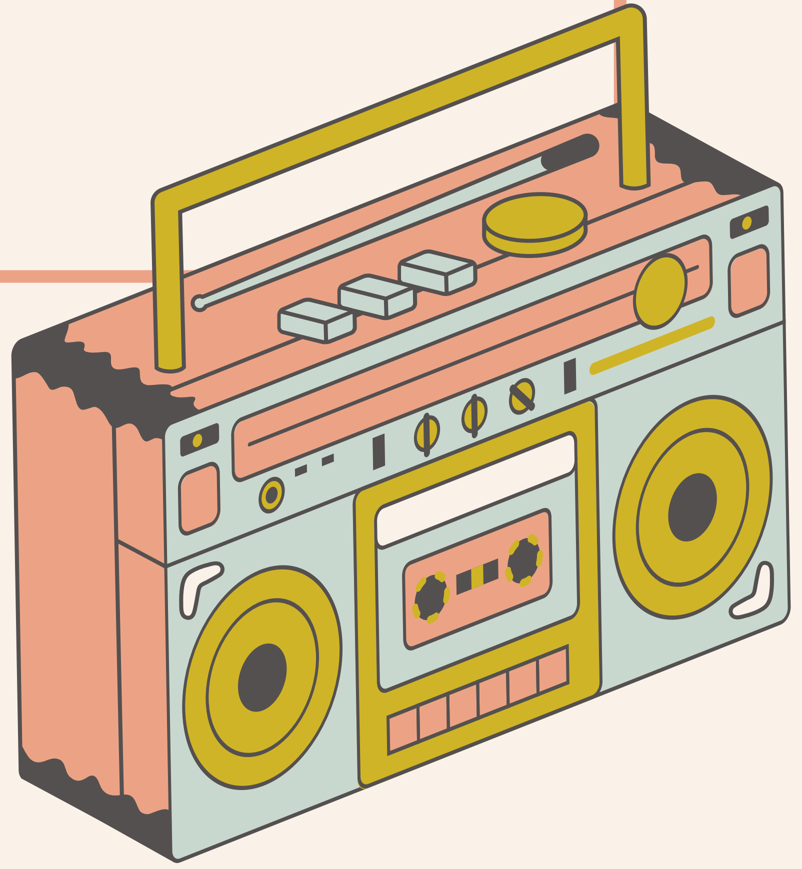
```
number= [2,3,4,5,6]

root= [i**(1/2) for i in number]

print(root)
```

explanation:

these was another way to write list comprehension. both way will make the same output.



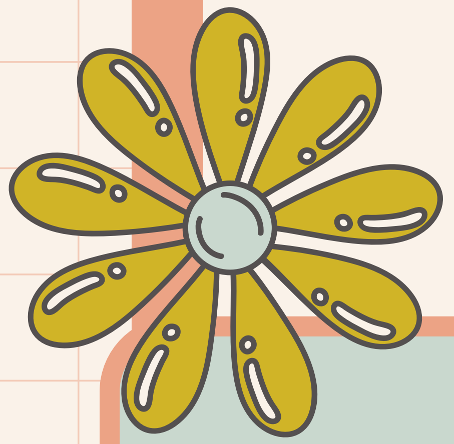
Error in python



there is 3 kind of error in python: syntax error, exception and logical error

note: some source say logical error and exeption error is same

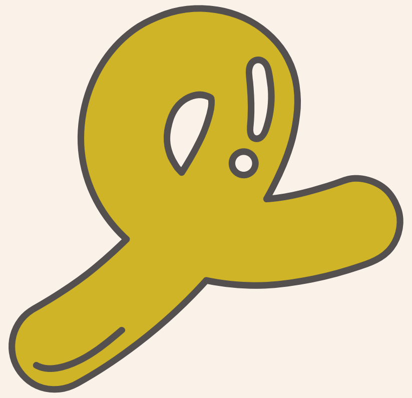
Syntax errors are similar to grammar or spelling errors in a Language. If there is such an error in your code, Python cannot start to execute your code. You get a clear error message stating what is wrong and what needs to be fixed. Therefore, it is the easiest error type you can fix



SYNTAX ERROR

SYNTAX ERROR DETAIL

Missing symbols (such as comma, bracket, colon), misspelling a keyword, having incorrect indentation are common syntax errors in Python



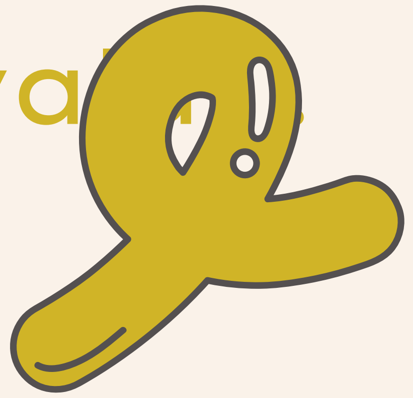
- Exceptions may occur in syntactically correct code blocks at run time. When Python cannot execute the requested action, it terminates the code and raises an error message.

EXCEPTION ERROR

EXCEPTION ERROR DETAIL

exception error consist of a lot type of error:

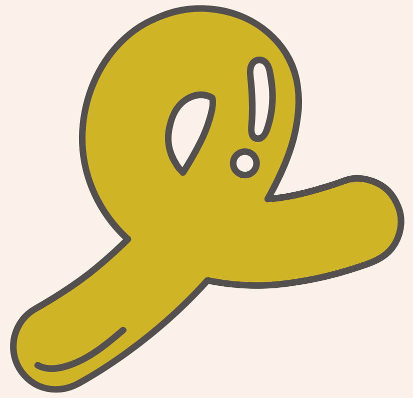
- ZeroDivisionError is occur when we try to divided any number with zero
- NameError is occur when variable is not found in local or global scope.
- ValueError is occur when a function gets an argument of correct type but improper va



CONTINUATION OF EXCEPTION ERROR DETAIL

exception error consist of a lot type of error:

- KeyError is occur when the index of a sequence is out of range.
- FileNotFoundError is occour when file we want to import not found
- TypeError is occour when a function or operation is applied to an object of incorrect type.
- and many other



LOGICAL ERROR

- Logical errors are the most difficult errors to fix as they don’t crash your code and you don’t get any error message. so, logical error occour when we dont get the result we want but the program still running

Example

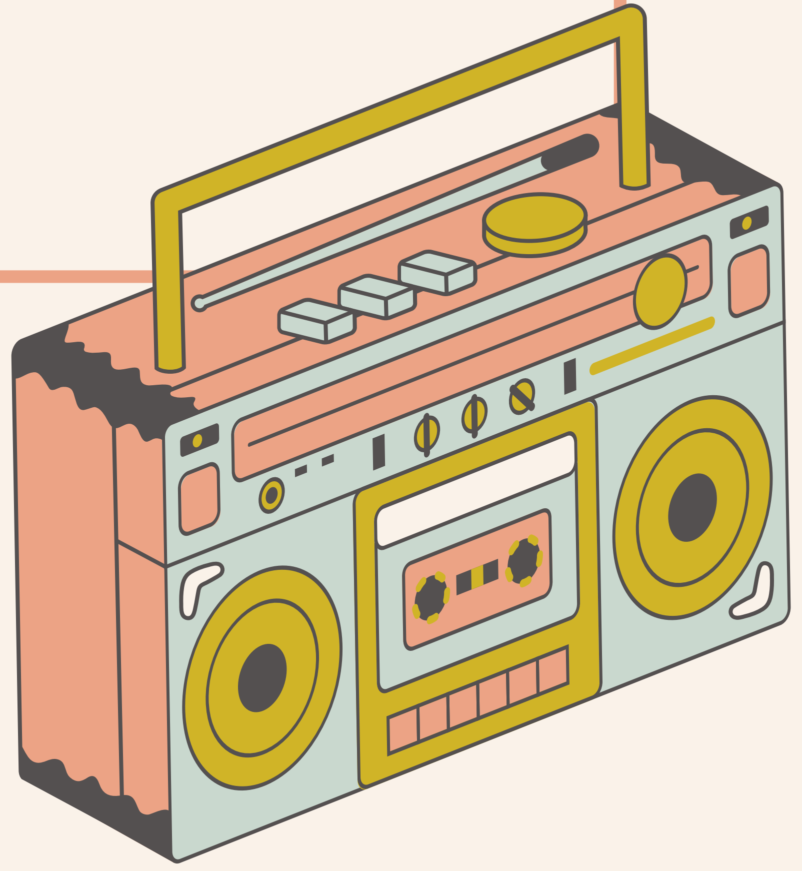
```
a=float(input("input first number= "))
b=float(input("input second number= "))

print("output formula (a+b)*3 equal to ",a+b*3)
print("expected output formula (a+b)*3",(a+b)*3)
```

output

```
input first number= 2
input second number= 3
output formula (a+b)*3 equal to 11.0
expected output formula (a+b)*3 15.0)
```

EXAMPLE LOGICAL ERROR

How to handle error in python



handling error can use try and except to make program keep running even there error. this way handling error need to know error name that will occur if we want to handle spesific error or make different output for different error. if we not fill error name means all error will be handle with same way

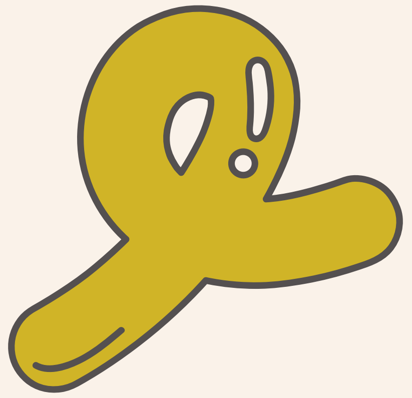
EXAMPLE HANDLING ERROR WITHOUT SPESIFIC NAME

```
try:
    print(x)
except:
    print("An Error or something went wrong!")
```

output:

```
an error or something when wrong
```

note:for this type handling any error will have same output

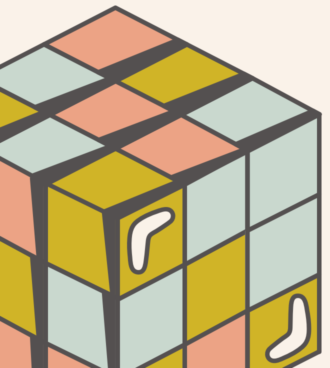


EXAMPLE HANDLING ERROR

```
a=0

try:
  bagi=1/a
  print(bagi)
except ZeroDivisionError:
  print("can't divide a number by zero")
```

output

```
can't divide a number by zero
```



CAN YOU GET THEM RIGHT?

Crown the winner!