**Booleans & Conditionals**

jueves, 9 de marzo de 2023

7:16 p. m.

Source: [Python Tutorial for Beginners 6: Conditionals and Booleans - If, Else, and Elif Statements](https://www.youtube.com/watch?v=DZwmZ8Usvnk)

Documentation: Booleans - <https://docs.python.org/3/c-api/bool.html?highlight=booleans>

Control flow (4.1 & 4.2) - <https://docs.python.org/3/tutorial/controlflow.html?highlight=conditional%20expression>

Else without if - [Python Tutorial: Else Clauses on Loops](https://www.youtube.com/watch?v=Dh-0lAyc3Bc&list=PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU&index=61)

Booleans

Just for reference

Comparisons

# Equal: ==

# Not Equal: !=

# Greater Than: >

# Less Than: <

# Greater or Equal: >=

# Less or Equal: <=

# Object Identity: is

False Values

# False

# None

# Zero of any numeric type

# Any empty sequence. For example, '', (), [].

# Any empty mapping. For example, {}.

Everything else in python evaluates to True

One thing to note is when comparing equality ( == ) operator y different from identity operator ( is ). The first will check if the value of the items are the same, but the later will check if they're located in the same space in memory.

Control Flow

* The elif keyword exist to avoid excessive indentation.

'The Switch Case'

Python does not have 'switch case' statements. But the common turnaround is to define the cases of the switch statement in a dictionary and define a function retuning .get() of that dict and a default value.

dic = {

    1: "one",

    2: "two",

    3: "three"

}

def switch(value):

    return dic.get(value, "Not Found")

print(switch(int(input('input number:')))) =

input number:1

one

input number:5

Not Found

Match Statements

Example: [Python 10 Match Case Statements | structural pattern matching](https://www.youtube.com/watch?v=dFfI6swA7co)

Source: (4.6) <https://docs.python.org/3/tutorial/controlflow.html?highlight=conditional%20expression>

PEP636 <https://peps.python.org/pep-0636/>

The Match statement is another way to simulate the switch case from other languages.

Is good to note that this statement only match once, meaning that once it founds a match it stops trying to find the next one.

def size\_match(size):

    match size:

        case 32:

            print("Slim")

        case 36:

            print('Medium')

        case 42:

            print('Thick')

size\_match(32) = Slim

Is possible to group up cases with the | (“or”) operator.

def size\_match(size):

    match size:

        case 32 | 34:

            print("Slim")

        case 36 | 38:

            print('Medium')

        case 40 | 42:

            print('Thick')

size\_match(38) = Medium

The statement also operates with bind variables (Like dicts, list or tuples).

def point\_match(point):

    match point:

        case (0, 0):

            print("In origin")

        case (value, 0):

            print('In x-axis')

        case (0, value):

            print('In y-axis')

point\_match((0,5)) = In y-axis

point\_match((8,0)) = In x-axis

point\_match(('x',0)) = In x-axis

The problem here is that this statement does not consider the date structure of the input, it only verifies if there is "a something" to match with this kind of syntax.

Now, a correction could be done like this

        case (int(), 0):

            print('In x-axis')

point\_match((8,0)) = In x-axis

point\_match(('x',0)) = Does not print anything. Match statement does not raise errors.

Since the statement doesn't raise any errors, there is a way to catch exceptions, adding a special case with just an underscore ( \_ ) and it will serve as a wildcard for any mismatch.

case \_:

            print(f'Match not found')

point\_match(('y','y')) = Match not found

The statement also works with conditionals.

def grade\_match(grade):

    match grade:

        case grade if grade >= 90:

            print("A")

        case grade if grade >= 80:

            print("B")

        case grade if grade >= 70:

            print("C")

        case grade if grade >= 60:

            print("D")

        case \_:

            print("F")

grade\_match(95) = A

grade\_match(72) = C

This is a case with membership.

def poketype\_match(poketype):

    match poketype:

        case poketype if poketype in ['Grass', 'Ice', 'Bug', 'Steel']:

            print("Vulnerable to Fire type")

        case poketype if poketype in ['Fire', 'Ground', 'Rock']:

            print("Vulnerable to Water type")

        case poketype:

            print(f'{poketype} type not updated yet')

This last case here is to show that the statement catch the input and can be delivered into an output

poketype\_match('Ice') = Vulnerable to Fire type

poketype\_match('Ghost') = Ghost type not updated yet

This is a clip I found over the true purpose of adding match statement.

[The real purpose of Python's match statement, feat. CSTs](https://www.youtube.com/watch?v=ASRqxDGutpA)

GAME CHANGER 
match node: 
case 