Untitled

August 26, 2019

Little Bob loves chocolate, and he goes to a store with Rs. N in his pocket. The price of each chocolate is Rs. C. The store offers a discount: for every M wrappers he gives to the store, he gets one chocolate for free. This offer shall be availed only once. How many chocolates does Bob get to eat?

```
In [1]: n=int(input("Enter the money in Hand: "))
```

```
Enter the money in Hand: 100
```

```
In [2]: c=int(input("Enter the Unit Price of Chocolate: "))
```

```
Enter the Unit Price of Chocolate: 5
```

```
In [3]: m=int(input("Enter Strore offer: "))
```

```
Enter Strore offer: 4
```

```
In [4]: print(n , c, m)
```

```
100 5 4
```

```
In [5]: p = n/c
```

```
In [6]: print(p)
```

```
20.0
```

```
In [7]: f = p/m
```

```
In [8]: print(f)
```

```
5.0
```

```
In [9]: print("Number of chocolates:: ", p+f)
```

```
Number of chocolates::  25.0
```

```
In [10]: set('spam'), 1,2,3,4
Out[10]: ({'a', 'm', 'p', 's'}, 1, 2, 3, 4)
In [11]: bool(f)
Out[11]: True
In [12]: import fractions
In [13]: x = fractions.Fraction(1,2)
In [14]: print(x)
1/2
In [15]: import decimal
In [16]: y = decimal.Decimal('1.0')
In [17]: print(y)
1.0
In [18]: complex(y)
Out[18]: (1+0j)
In [19]: y.real
Out[19]: Decimal('1.0')
In [20]: y.imag
Out[20]: Decimal('0')
In [21]: type(n)
Out[21]: int
In [22]: type(p)
Out[22]: float
In [1]: str1 = 'VIT'
In [2]: str2 = "VIT"
```

```
In [3]: str1 == str2

Out[3]: True

In [5]: type(str1)

Out[5]: str

In [6]: type(str2)

Out[6]: str

In [7]: "hello" + "world"

Out[7]: 'helloworld'

In [8]: "hello" + 3

        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-8-295c36e74c04> in <module>()
    ----> 1 "hello" + 3


        TypeError: must be str, not int

In [9]: "hello" * "world"

        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-9-2df74e4a2a45> in <module>()
    ----> 1 "hello" * "world"


        TypeError: can't multiply sequence by non-int of type 'str'

In [10]: 0.5 + 3

Out[10]: 3.5

In [11]: 0.5 * 3
```

```
Out[11]: 1.5

In [12]: a = 1

In [13]: b = 2

In [14]: a

Out[14]: 1

In [15]: b

Out[15]: 2

In [16]: a = 2

In [17]: a

Out[17]: 2

In [18]: a = 1

In [19]: b = a

In [20]: a = 5

In [21]: a

Out[21]: 5

In [22]: b

Out[22]: 1

In [23]: hello

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-23-f572d396fae9> in <module>()
----> 1 hello


NameError: name 'hello' is not defined
```

Variables are simple to understand, but there are a few details that we need to keep in mind:

Variables always point to a value, they never point to other variables. That's why the arrows in our diagrams always go left to right.

Multiple variables can point to the same value, but one variable cannot point to multiple values.

The values that variables point to can point to other values also. We'll learn more about that when we'll talk about lists

```
In [24]: if = 123


          File "<ipython-input-24-949a24583bb3>", line 1
        if = 123
           ^
    SyntaxError: invalid syntax
```

```
In [28]: a = 1
         a = a + 1

In [29]: a

Out[29]: 2

In [30]: a += 2          # a = a + 2
         print(a)
         a -= 2          # a = a - 2
         print(a)
         a *= 2          # a = a * 2
         print(a)
         a /= 2          # a = a / 2
         print(a)

4
2
4
2.0
```

This is not limited to integers.

```
In [31]: a = 'hello'
         print(a)
         a *= 3
         print(a)
         a += 'world'
         print(a)

hello
hellohellohello
hellohellohelloworld

In [32]: hello = 'hello there'

In [33]: print(hello)
```

```
hello there
```

```
In [34]: a = hello
```

```
In [35]: a
```

```
Out[35]: 'hello there'
```

```
In [37]: print('Hello')
```

```
Hello
```

```
In [38]: print('Hello")
```

```
          File "<ipython-input-38-0779651877b1>", line 1
        print('Hello")
                      ^
    SyntaxError: EOL while scanning string literal
```

```
In [39]: print("Hello')
```

```
          File "<ipython-input-39-21f0c5b97628>", line 1
        print("Hello')
                      ^
    SyntaxError: EOL while scanning string literal
```

```
In [40]: print('Let's go')
```

```
          File "<ipython-input-40-2d1e7b12ed8a>", line 1
        print('Let's go')
                   ^
    SyntaxError: invalid syntax
```

```
In [42]: print("Let's Go!")
```

```
Let's Go!
```

```
In [43]: 0.1024
```

```
Out[43]: 0.1024

In [44]: .1024

Out[44]: 0.1024

In [48]: print(bool(True))
         print(bool(False))
         print(bool("VIT"))
         print(bool(""))
         print(bool(''))
         print(bool(0))
         print(bool(3))
         print(bool(None))
         print(True + 25)
         print(False + 25)
         print(bool.__bases__)
         print(True + True)

True
False
True
False
False
False
True
False
26
25
(<class 'int'>,)
2


In [49]: a = 3; b = 4
         (a < b) * 10 + (a == b) * 20

Out[49]: 10

In [50]: x = None
         help(x)

Help on NoneType object:

class NoneType(object)
 |  Methods defined here:
 |
 |  __bool__(self, /)
 |      self != 0
 |
```

```
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  __repr__(self, /)
 |      Return repr(self).
```

```
In [51]: 0b10000

Out[51]: 16

In [52]: 0o20

Out[52]: 16

In [53]: 0x10

Out[53]: 16

In [55]: oct(16)

Out[55]: '0o20'

In [57]: hex(16)

Out[57]: '0x10'

In [58]: bin(16)

Out[58]: '0b10000'

In [83]: thisset = set('spam'), 1,2,3,4

In [80]: type(thisset)

Out[80]: tuple

In [84]: thisset[0]

Out[84]: {'a', 'm', 'p', 's'}

In [85]: thisset[-1]

Out[85]: 4

In [87]: 3.0-3.0

Out[87]: 0.0

In [88]: 1.0e-300 / 1.0e100
```

```
Out[88]: 0.0

In [89]: 1.0e-300 / 1.0e100 == 3.0 -3.0

Out[89]: True
```

Repeated Print

```
In [90]: print('a' *15)

aaaaaaaaaaaaaaa


In [91]: print('\na' *15)


a
a
a
a
a
a
a
a
a
a
a
a
a
a
a


In [97]: comp_num1 = complex(3,4)

In [98]: comp_num1

Out[98]: (3+4j)

In [99]: comp_num1.real

Out[99]: 3.0

In [100]: comp_num1.imag

Out[100]: 4.0

In [101]: comp_num2 = complex(3,-2)

In [102]: comp_num1 + comp_num2
```

```
Out[102]: (6+2j)

In [103]: comp_num1 * comp_num2

Out[103]: (17+6j)

In [104]: a, b, c, d = 'spam'

In [105]: a

Out[105]: 's'

In [106]: b

Out[106]: 'p'

In [107]: c

Out[107]: 'a'

In [108]: d

Out[108]: 'm'
```

Extended Unpacking

```
In [113]: a, *b = 'spamer'

In [114]: a

Out[114]: 's'

In [115]: b

Out[115]: ['p', 'a', 'm', 'e', 'r']

In [116]: spam = ham = 'lunch'

In [117]: spam

Out[117]: 'lunch'

In [118]: ham

Out[118]: 'lunch'

In [122]: spam += ' over'

In [123]: spam

Out[123]: 'lunch over'

In [125]: a,b,c = range(1,4)
```

```
In [126]: a

Out[126]: 1

In [127]: b

Out[127]: 2

In [128]: c

Out[128]: 3
```

Swap

```
In [129]: nudge = 1
          wink = 2
          nudge, wink = wink, nudge

In [130]: nudge

Out[130]: 2

In [131]: wink

Out[131]: 1
```