IT1007 INTRODUCTION TO PROGRAMMING

Python and C

Variables

- For storage of data

```
>>>  x  =  3
```

This space is allocated and labeled as “x”. And we store ‘3’ inside

Computer Memory

x

Variable Naming

- Start with 'a'-'z' or 'A'-'Z' or '_'
- Contain only alphanumeric characters or '_'
- Case sensitive

```
X_1 != x_1
```

- Avoid reserved keywords e.g. `if`
- Python convention: lower case letters separated by '_'
  - e.g. `count_change`

Variable Types

8, 45, 123	int
2.71828, 3.14159 , 1.0	float
True, False	bool
“it1007” ‘it1007’	str
	None

Variable Type

```
>>> x = 3
>>> name = “Alan”
```

This space is allocated and labeled as “x”. And we store ‘3’ inside. And can store integers only

This space is allocated and labeled as “name”. And we store “Alan” inside. And can store strings only

The function Type(…)

```
>>> type(123)
<class 'int'>

>>> type('123')
<class 'str'>

>>> type(None)
<class 'None'>
```

Type conversion

```
>>> str(123)
'123'

>>> float('45.2')
45.2

>>> int(23.8)
23

>>> int('cs1010s')
ValueError!
```

Assignments

Doesn’t matter if it’s quote or double quote

```
>>> abc = 18
>>> my_string = 'This is my string'
>>> x, y = 1, 2
```

Assignments

```
>>> x = 10
>>> x = 2
>>> x = 4
>>> print(x)

???
```

```
>>> a, b, c = 1, 2, 3
>>> a, b, c = c, b, a
>>> print(a, b, c)
???
```

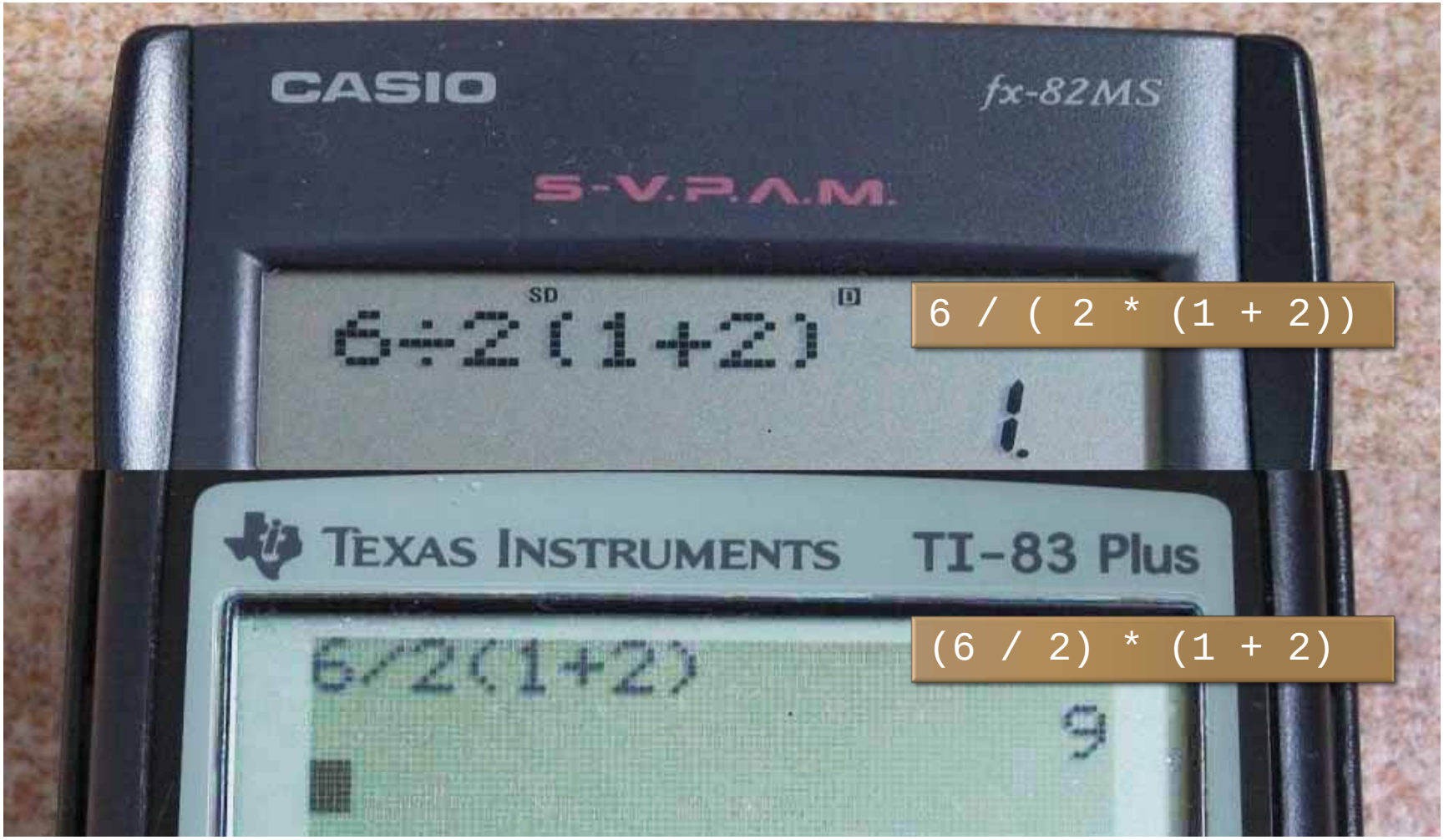
"=" is different from our usual "equal" in math

Arithmetic: + - * / ** // %

```
>>> a = 2 * 3
>>> a
6

>>> 2 ** 3
8
```

```
>>> 11 / 3
3.6666666666666665

>>> 11 // 3
3

>>> 11 % 3
2
```

Operator Precedence



Python Operator Precedence

- 6/2*(1+2)
- 3*(1+2)
- 3*(1+2)
- 3*3
- 9

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Boolean: Truth values

- Statements can be either True or False
- `2 > 1` is True
- `5 < 3` is False

Operators

- Comparison:

```
>>> 1 <= 10
True
>>> 5 > 15
False
>>> 5 <= 5
True
```

```
>>> 2 != 3
True
>>> '1' == 1
False
>>> False == False
True
>>> True != True
False
```

The very no. 1 trap for programmers

Operators

- Logic:

```
>>> True or False
True
>>> True and False
False
>>> not False
True
```

- a or b True if either a or b is True
- a and b True if both a and b are True
- not a True if a is not True

Truth Tables

A	NOT A
True	False
False	True

A OR B		A	
		True	False
B	True	True	True
	False	True	False

A AND B		A	
		True	False
B	True	True	False
	False	False	False

Truth Value Revisted

- Python has keywords `True` and `False`
- In Python 3.x, `True` and `False` will be equal to `1` and `0`
- Anything that is not `0` or empty will be evaluated as `True`
- Logic:

```
>>> True and 0
0
>>> not 'abc'
False
>>> 1 or 0
1
```

Strings

```
>>> s = 'ba'
>>> t = 'ck'
>>> s + t
'back'
>>> t = s + 'na' *
2
>>> t
'banana'
```

```
>>> 'z' in t
False
>>> 'bananb' > t
True
>>> 'banan' <= t
True
>>> 'c' < t
False
```

lexicographical ordering: first the first two letters are compared, and if they differ this determines the outcome of the comparison; if they are equal, the next two letters are compared, and so on, until either sequence is exhausted.

Strings

```
>>> w = ‘banana’
>>> s = w + w
>>> print(s)
‘bananabanana’
>>> s = w*3
>>> print(s)
‘bananabananabanana’
```

```
>>> s = (w+’ ‘)*2
>>> print(s)
‘banana banana ‘
```

Strings

- A String is a sequence of characters
- We can index a string, i.e.

```
>>> s = 'abcd'
>>> s[0]
'a'

>>> s[2]
'c'
```

- The index of the first character is 0

String Slicing

Non-inclusive

Default
start = 0
stop = #letters
step = 1

```
s[start:stop:step]
```

```
>>> s = 'abcdef'
>>> s[0:2]
'ab'

>>> s[1:2]
'b'

>>> s[:2]
'ab'
```

```
>>> s[1:5:3]
'be'

>>> s[::2]
'ace'

>>> s[::-1]
???
```

Strings

```
c = "#"
s = " “

print(" ")
print("   *")
print(s*3 + c)
print(s*2 + c * 3)
print(s + c * 5)
print(s*3 + c)
print(s*2 + c * 3)
print(s + c * 5)
print(c * 7)
print(s*3 + c)
```

```
  *
  #
 ###
#####
  #
 ###
#####
#######
  #
```

ASCII Art

```
   _         _
  (c).-.(c)
   /  ._.  \
 __\(  Y  )/__
(_.-/'-'\-._)
   || X ||
  _.'  `-'  '._
 (.-./`-'\.-.)
  `-'       `-'
```

```
   @~t@
  @~~t@  @@@
  @~(t@ %^^^^@
  @((tt@s^//^@
   @(tt@s////@
   @ttC@s^^/^@
   @CCC@tts^s@ @@@G/////@@
  @O~CC@%ttst@@    /(~~~//@
 @O   ///((((/^     ~~~~~~///@
@O   //(@@@(((^       ~~~//(///@
@O^  /(@ @@(((^       ~~~/~///@
@^^^^/(@@@(((/^        ~(~~/t(/@
@^^   ((((((/^^        (~~~/t((@(@
@  ^   ((((/^^        ((~~t/t((@(@
 @ s    ^^^^^         (t~%ttt((C@@
      @%%             ((((((%%%@
      @  %s          (((((%%%%%@
    @  t      @ @  OO%%%%%%%%%@
      @@@@@@  @@@@@@@@@@@@@@@@
```

Admin

- You should have installed your Python and Lab 00

- We have a Facebook Page
  - https://www.facebook.com/groups/101073953956082/
  - Or search for IT1007 NUS 2017