Section A1 Fundamentals

How to execute your cell

Using a *shortcut*

- **Ctrl + Enter** run selected cell
- **Shift + Enter** run the current cell, move to the next cell below
- **Alt + Enter** run the current cell, insert a new cell below

#To look up what Print function does (using ? symbol). This can be applied to any python function
?print

Variable&Types

A variable:

- can have a short name e.g. x or more descriptive name
  e.g. weight or lastName
- can contain letters (A-z), numbers (0-9) and underscores (_)
- cannot start with a number
- is case-sensitive (which means weight is a different variable
  than Weight).

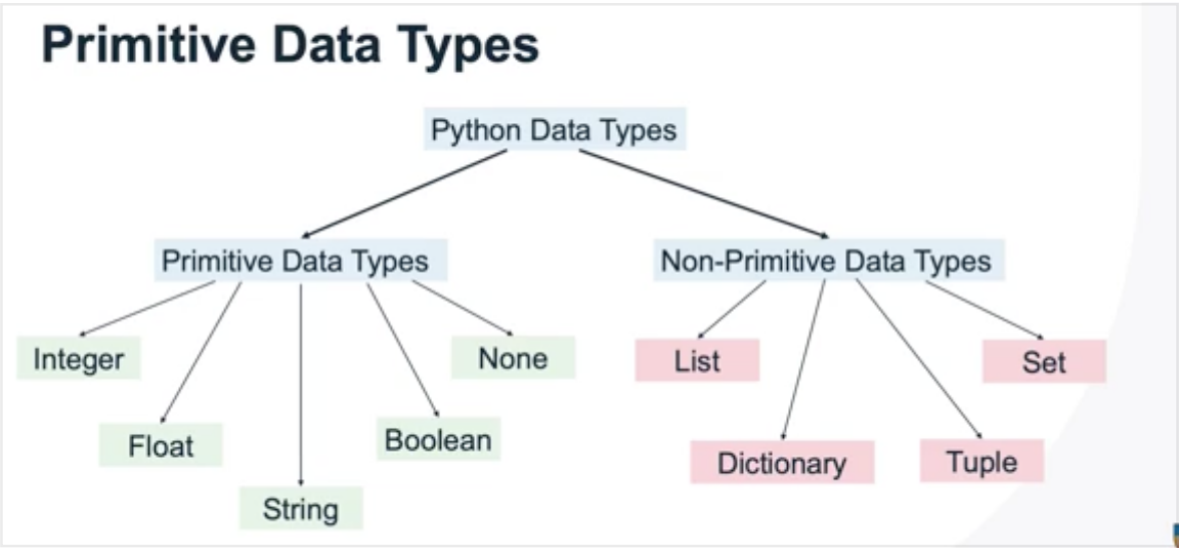
You can give any name to your variable. However, a reserved word cannot be used as a variable name. Here is a list of reserved words in Python.

and	del	from	None	try
as	elif	global	nonlocal	True
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	return
def	for	lambda		

- Every value has a type
- `type()` to check the type of data stored in a variable

Python has **several built-in data types** in the following categories to store **different kinds of information** in variables.

Text type	str
Numerical type	int, float, complex
Sequence type	list, tuple, range
Mapping type	dict
Set type	set, frozenset
Boolean type	bool
Binary type	bytes, bytearray, memoryview



Numbers

An **integer number** does not have size restrictions; it can be arbitrarily large. A difference from integer numbers is that a decimal number has a limit of digits of precision (e.g. the number of digits after the decimal). This is because, with less precision, the calculations run faster. You just need to be aware of this in case your data needs very high precision.

**	Exponentiation
//	Integer division
%	Modulo (division remainder)

//：truncate the decimal part and keep the integer

Unary Operators and Operator Precedence 一元运算符和运算符优先级
-：正变负负变正 +：无影响

Order	**Operator**
First	Exponentiation **

Second	Multiplication *, division (/ and //) and modulo %
Third	Addition + and subtraction -

Associativity indicates which operation will happen first when you have two operators with the same precedence in an expression. Operators associate from left to right.

dir() function
help (math.pow)

random.random() ： The basic random function is good for probabilities since it gives a number between 0.0 and 1.0 (inclusive)
random.uniform()： This function needs a starting point (the minimum number allowed) and stopping point (the maximum number allowed)
random.randint()： This function needs a starting point (minimum number allowed) and stopping point (maximum number allowed).

Strings

A triple quotation is used for strings that cover more than one line.

use the backslash \ to get certain special characters (called escaped characters) into your string.

```
#Repeating strings for n times
str1="Hello! "
str1*3
```

get input from the user by using the input( ) function

```
In [14]: name = input("Enter your name: ")
         print("You entered", name)
Out[14]: Enter your name: David
         You entered David
```

Strings have a lot of built-in methods

Method	Description
`lower()`	Returns a string with all the letters of the original string in lowercase
`upper()`	Returns a string with all the letters of the original string in uppercase
`replace(x,y)`	Returns a string with every occurrence of x replaced by y
`count(x)`	Counts the number of occurrences of x in the string
`isalpha()`	Returns **True** if all the characters of the string are letters otherwise **False**

lower(), upper() and replace(x,y) methods do not change the original string.

In Python, we use the **True** and **False** keywords to define boolean variables. It is important to note that the first letter in the keywords True and False must be Upper Case

we use the **None** keyword to define a null value

`None` is not the same as zero.
`None` is not the same as an empty string.
`None` is not the same as `False`.
When you compare `None` to anything, it will always return `False` (except `None` itself).

Type Converting

int(), float() or str()

#Converting a number to a boolean variable
All non-zero values are True!!!!

```
#Converting a number to a boolean variable
# All non-zero values are True!!!!

intNum = 0 #integer number
floatNum = 1.0 #Float number
intBig = -45

intBigBool = bool(intBig)
intbool = bool(intNum) # Converting a integer number to a boolean
floatbool = bool(floatNum) # Converting a float number to a boolean

print('intbool:  ', intbool,' Type', type(intbool))
print('intbigbool:  ', intBigBool,' Type', type(intBigBool))
print('floatbool:', floatbool,' Type', type(floatbool))
```

Quiz

Question 22 5 pts

Markdown supports including images as shown in the example below.

Select the correct Markdown syntax to include the above image taken from https://www.python.org/static/community_logos/python-logo-master-v3-TM.png. When a user clicks the above image, it should go to the following URL: https://www.python.org/ .

- [x] `[![](https://www.python.org/static/community_logos/python-logo-master-v3-TM.png "Python logo")](https://www.python.org/)`
- [x] `[![python logo](https://www.python.org/static/community_logos/python-logo-master-v3-TM.png "Python logo")](https://www.python.org/)`
- [ ] `!{https://www.python.org/static/community_logos/python-logo-master-v3-TM.png}`
- [ ] `[![python logo](https://www.python.org/static/community_logos/python-logo-master-v3-TM.png "Python logo")][https://www.python.org/]`
- [x] `![](https://www.python.org/static/community_logos/python-logo-master-v3-TM.png)`

W2

Logical Expressions

`not a`	True if a is False (returns opposite of a)

Debugging with the scientific method

1. ***Observe***
2. ***Hypothesise***
3. ***Predict***
4. ***Test***
5. ***Conclude***

Assert

```
assert condition, message
```

It takes two parameters.

- *condition:* This is the condition that you want to test in your program (required)
- *message:* This is the message that you want to display when the condition is False (optional)

- ***SyntaxError:*** When the parser encounters a syntax error.
- ***NameError:*** When a name is not found.
- ***TypeError:*** When an operation is applied to an inappropriate type
- ***ValueError:*** When an operation or function receives an argument that has the right type but an inappropriate value.
- ***ImportError:*** When the import statement has troubles trying to load a module
- ***IndentationError:*** For syntax errors related to incorrect indentation
- ***ZeroDivisionError:*** When the second argument of a division or modulo operation is zero.
- ***AttributeError:*** When an attribute reference or assignment fails.
- ***AssertionError:*** When an assert statement fails.

Debugging using logging module.

Level	When to Use
DEBUG	Detailed information, typically of interest only when diagnosing problems.
INFO	Confirmation that things are working as expected.
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

the default logger is the ‘root’ and its default **basicConfig** level is set to WARNING, meaning that only messages from logging.warning() and higher

levels will get logged.

What if we want to print DEBUG and INFO messages as well? To do that, you will have to set basicConfig level to DEBUG