Intro to Scientific Programming to:

python

- Day 1

About me

Claudio Mirabello, Ph.D. (claudio.mirabello@liu.se)

- Principal Research Engineer, bioinformatics division
- Work at National Bioinformatics Infrastructure Sweden
- Background as computer engineer
- Expertise: machine learning, data science
- Italian 🤌
- Food motivated

About me

Claudio Mirabello, Ph.D.

- Visiting address: B-huset 2B:586
- E-mail: claudio.mirabello@liu.se
- Fridays 10-11AM only

About the course

- Course website: https://github.com/clami66/workshop-python/tree/0422/ (https://github.com/clami66/workshop-python/tree/0422/)
- Based on NBIS course for bioinformatics (https://github.com/NBISweden/workshop-python/)
- Two lectures per week, for a total of 3 hours
- Who knows how much we'll cover!
- One final project applying what you have learned on your research
- Please make sure you have installed conda and jupyter notebook
- You can initially use google colab (https://colab.research.google.com/) if you have issues

What is programming?

Wikipedia:

"Computer programming is the process of building and designing an executable computer program for accomplishing a specific computing task"

What can we use it for?

Endless possibilities!

- convert/parse data files
- compute complex equations from raw measurements
- fit curves to your data
- plotting of results
- automatize your stuff -> no more manual mistakes!

Course content

- Core concepts about Python syntax: Data types, blocks and indentation, variable scoping, iteration, functions, methods and arguments
- Different ways to control program flow using loops and conditional tests
- Writing functions and best-practice ways of making them usable
- Reading from and writing to files
- (Regular expressions and pattern matching)
- Using extra Python libraries:
    - Numpy for scientific computing
    - Matplotlib for plotting
    - pandas for table handling

Learning outcomes

After this course you should be able to:

- Describe and apply basic concepts in Python, such as:
    - Loops
    - If/else statements
    - Functions
    - Reading/writing to files
- Being able to edit and run Python code
- Write file-processing Python programs that produce output to the terminal and/or external files
- Create stand-alone python programs to process your data
- Know how to develop your skills in Python after the course (including debugging)

Some good advice

- 5 weeks to learn Python is not much
- Amount of information will decrease over days
- Complexity of tasks will increase over days
- Read the error messages!
- Save all your code

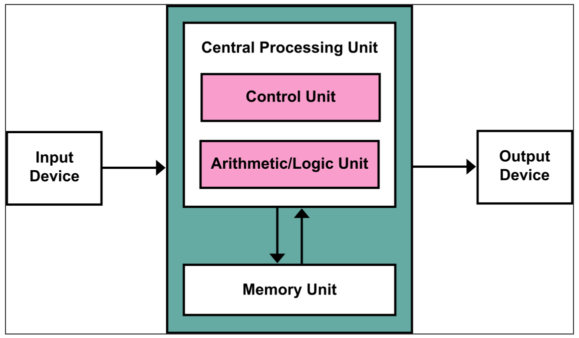
<u>How to seek help:</u>

- Google
- Ask your neighbour
- Ask me (slow response time)

Day 1

- Computer architecture
- Types and variables
- Operations
- Loops
- if/else statements
- How to: jupyter notebooks

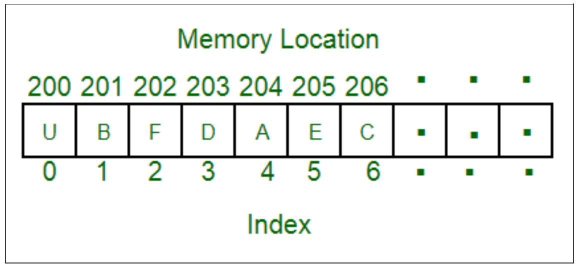
Computer Architecture



(src: wikipedia (https://en.wikipedia.org/wiki/Von_Neumann_architecture))

Computer architecture

- Very important to keep in mind how memory (RAM) works



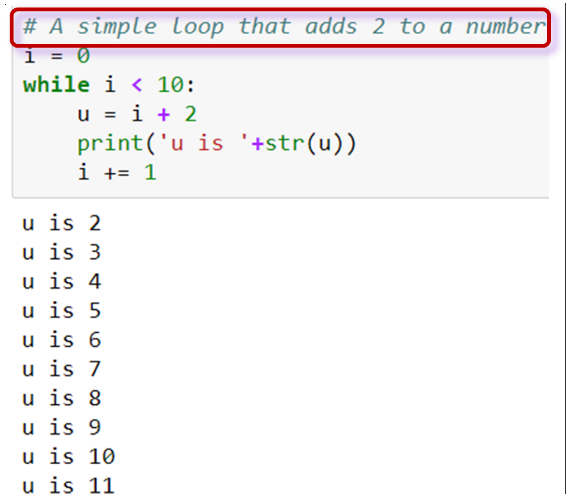
(src: Geeksforgeeks (https://www.geeksforgeeks.org/how-to-copy-elements-of-an-array-in-a-vector-in-c/))

Example of a simple Python script

In [ ]:

```python
# A simple loop that adds 2 to a number
i = 0
while i < 10:
    u = i + 2
    print('u is' + str(u))
    i += 1
```

Example of a simple Python script



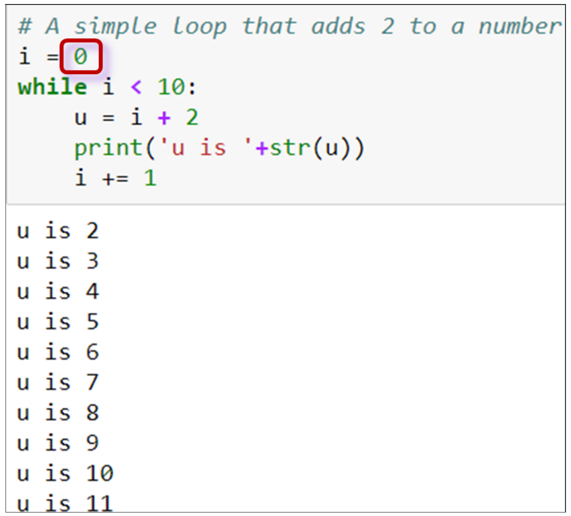
```
# A simple loop that adds 2 to a number
i = 0
while i < 10:
    u = i + 2
    print('u is '+str(u))
    i += 1
```

```
u is 2
u is 3
u is 4
u is 5
u is 6
u is 7
u is 8
u is 9
u is 10
u is 11
```

Comment

All lines starting with # is interpreted by python as a comment and are not executed. Comments are important for documenting code and considered good practise when doing all types of programming

Example of a simple Python script



```
# A simple loop that adds 2 to a number
i = 0
while i < 10:
    u = i + 2
    print('u is '+str(u))
    i += 1
```

```
u is 2
u is 3
u is 4
u is 5
u is 6
u is 7
u is 8
u is 9
u is 10
u is 11
```

Literals

All literals have a type:

• Strings (str)	‘Hello’ “Hi”
• Integers (int)	5
• Floats (float)	3.14
• Boolean (bool)	True or False

Literals define values

In [2]:
```
'this is a string'
"this is also a string"
3       # here we can put a comment so we know that this is an integer
3.14    # this is a float
True    # this is a boolean

type('this is a string')
type(2)
```

Out[2]:
```
int
```

Collections

In [3]:
```
[3, 5, 7, 4, 99]       # this is a list of integers

('a', 'b', 'c', 'd')   # this is a tuple of strings
{'a', 'b', 'c'}        # this is a set of strings
{'a':3, 'b':5, 'c':7}  # this is a dictionary with strings as keys and integers as values

type([3, 5, 7, 4, 99])
```

Out[3]:
```
list
```

What operations can we do with different values?

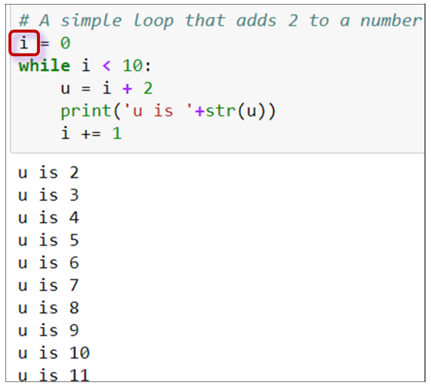
That depends on their type:

In [1]:
```
#'a string '+'          another string'
#2 + 3.4
'a string ' * 3
#'a string ' * 3.4
2**2
res = "tokyo"
"The result is: " + res
```

Out[1]:
```
'The result is: tokyo'
```

Type	**Operations**
int	+ - / ** % // ...
float	+ - / * % // ...
string	+

Example of a simple Python script



```
# A simple loop that adds 2 to a number
i = 0
while i < 10:
    u = i + 2
    print('u is '+str(u))
    i += 1
```

```
u is 2
u is 3
u is 4
u is 5
u is 6
u is 7
u is 8
u is 9
u is 10
u is 11
```

Identifiers

Identifiers are used to identify a program element in the code.

For example:

- Variables
- Functions
- Modules
- Classes

Variables

Used to store values and to assign them a name.

Examples:

- `i       = 0`
- `counter = 5`
- `snpname = 'rs2315487'`
- `snplist = ['rs21354', 'rs214569']`

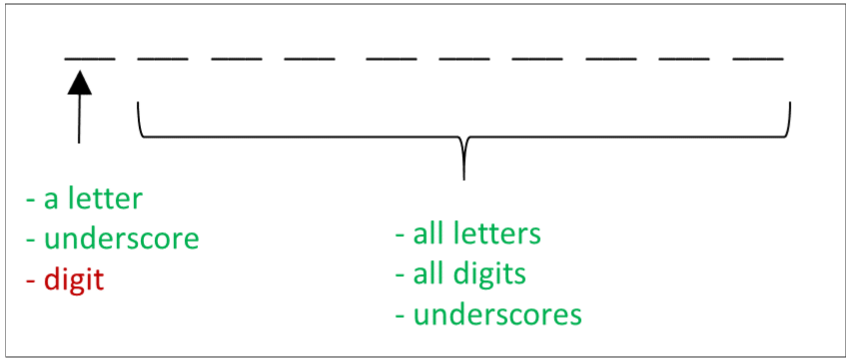
In [4]:
```
width  = 23564
height = 10

snpname = 'rs56483 '
snplist = ['rs12345','rs458782']

width * height
```

Out[4]:
```
235640
```

How to correctly name a variable

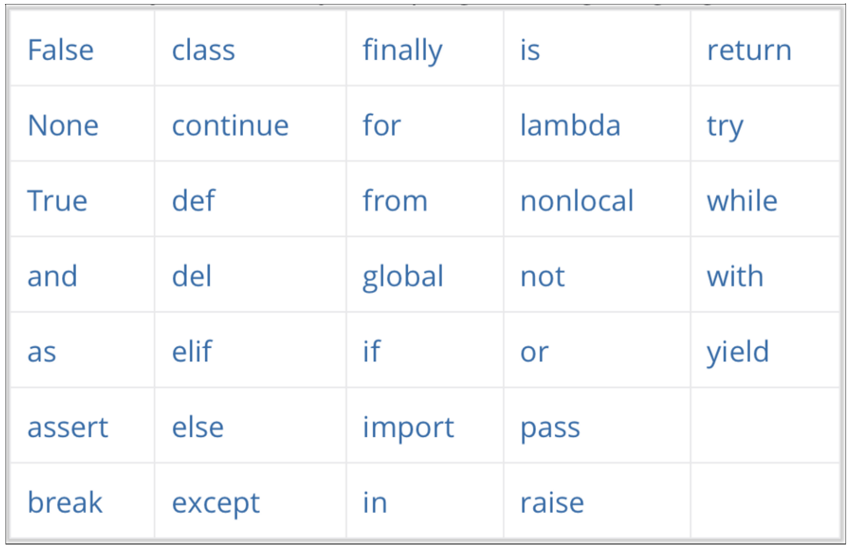


Allowed:
Var_name
_total
aReallyLongName
with_digit_2
dkfsjdsklut *(well, allowed, but NOT recommended)*

Not allowed:
2save
*important
Special%
With spaces

NO special characters:
+ - * $ % ; : , ? ! { } ( ) < > “ ‘ | \ / @

Reserved keywords



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

These words can not be used as variable names

Summary

- Comment your code!
- Literals define values and can have different types (strings, integers, floats, boolean)
- Values can be collected in lists, tuples, sets, and dictionaries
- The operation that can be performed on a certain value depends on the type
- Variables are identified by a name and are used to store a value or collections of values
- Name your variables using descriptive words without special characters and reserved keywords

NOTE!

How to get help?

- Google (https://www.google.com/) and Stack overflow (https://stackoverflow.com/) are your best friends!
- Official python documentation (https://docs.python.org/3/)
- Ask your neighbour
- Ask me