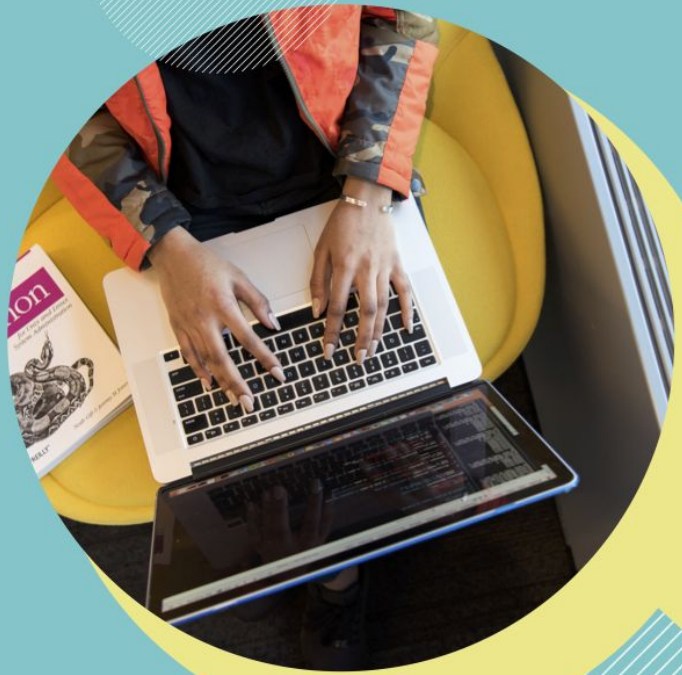
TECHNOMADS SEPTEMBER MEETUP

INTRO TO PYTHON

Monday 16th September, 6-8pm
Liverpool Science Park IC1

Join us for #nationalcodingweek!



Setup & Installation

We will be using python3 for this tutorial. Python 2 will no longer be supported from Jan 2020.

Python docs:
https://docs.python.org/3/using/index.html

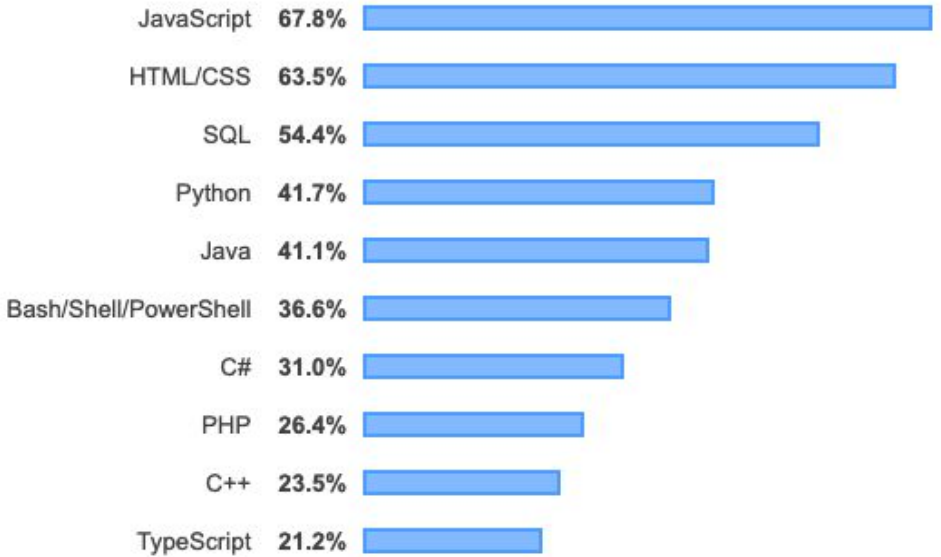
Anaconda:
https://www.anaconda.com/distribution/

Python in browser:
https://trinket.io/python3

Why learn python?

- General purpose and versatile
- Ideal first programming language
- Concise and easy to read
- Can be used for web development, software development and data science

4th most popular programming language from Stack Overflow, gaining popularity year on year



What would you like to use python for?

To get started with coding?

Web development?

Software development?

Data science?

Syntax

print() - Prints information to the console, or output device

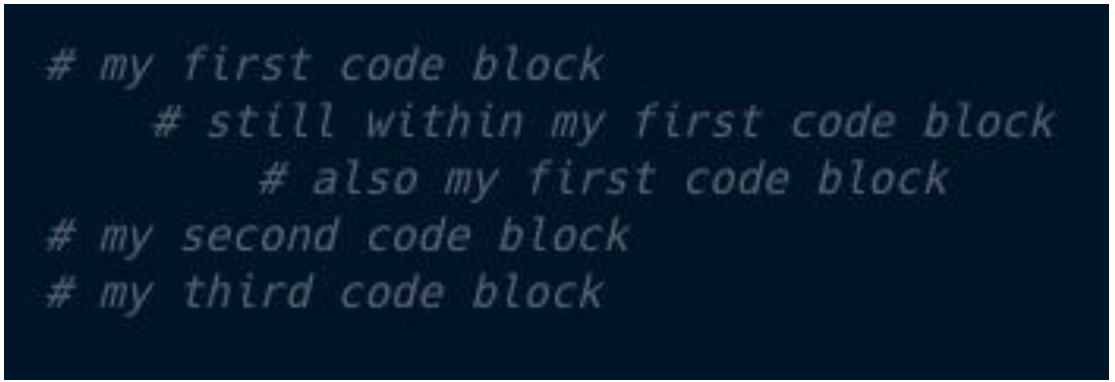
`print(object(s), separator=separator, end=end, file=file, flush=flush)`

```
print("Hello World!")

print(100)

pi = 3.14159
print(pi)

import math
print(math.sqrt(16))
```

Syntax

Indentation

Python programs are structured using indentation as shown here using comments



Syntax

Variables

Variable names can only include numbers, letters and underscores and cannot start with a number.
Supported data types include (for starters) - strings, integers, floating points, booleans

```
# These are all valid variable names and assignment
user_name = "jdoe"
userID = 100
prompt21 = "Enter your user name"
existing_user = False
another_userID = userID

# A variable's value can be changed after assignment
car_weight = 1000
car_weight = 2000

# This variable name is invalid
1entry = "Test"
```

Syntax

Operations

- `+` for addition
- `-` for subtraction
- `/` for division
- `*` for multiplication
- `%` for modulo
- `**` for exponential
- `+` for string concatenation
- `+=` for adding existing values (including string concatenation)

Syntax: Test your knowledge

0: Print Hello World!

1: Print your first name and surname separated by an underscore

2: Create variables containing your first name and surname, print!

3: Create a new variable concatenating your first and surname, print!

Control Flow

Relational operators:

- `==` checks if two values are equal
- `!=` checks if two values are not equal
- `>` checks if first value is greater than the second value
- `<` checks if first value is less than the second value
- `>=` checks if first value is greater than or equal to the second value
- `<=` checks if first value is less than or equal to the second value
- Boolean `and` `or` `not` can be used to compare multiple statement

This bring us to `if` `else` `elif` statements

```
x = 4
y = 5
if x < y:
    print('This statement passed!')
```

Control Flow: Test your knowledge

1: Write an if statement with condition

2: If statement pass - print statement

3: Otherwise, print a different statement

Bonus: create multiple conditions using `elif`

Lists

Ordered collection of items and can use multiple data types

```
numbers = [1, 2, 3, 4, 10]
names = ['Jenny', 'Alexis', 'Sam']
mixed = ['Jenny', 1, 2]
list_of_lists = [['a', 1], ['b', 2]]
```

- `.append()` used to add items to list
- `+` to add multiple lists
- `len()` to check the number of items in a list
- Zero indexing, access item with `myList[index]`
- Slice a list with `myList[START:END]` - can also leave blank or negative values
- `.count()` for the number of matching items
- `.sort()` to order the list OR `sorted(myList)` to return a new list, original list unmodified

Lists: Test your knowledge

1: Create a list of strings with at least 3 names

2: Print the second name in the list Hint: remember zero indexing

3: Add at least one new item to the list that already exists

4: Count the occurances of that item

5: Sort your list

Loops

- Must be indented
- Loop over items within a list with `for item in myList:`
- Loop over range with `for i in range(10):` or give start `range(start, end)`
- `break()` to escape the loop
- `continue()` to skip the rest of the code within the loop
- `while` loop executes code block while the statement is true
- Can use list comprehensions to create lists: `[EXPRESSION for ITEM in LIST <if CONDITIONAL>]`

Loops: Test your knowledge

1: Loop over your list and print each item

2: Create an if statement to break out your loop when a condition is met

Functions

To avoid rewriting code, can create functions to perform a series of operations and `return` a value

Function definitions may include parameters, providing data input to the function.

```
# define a function called
# my_function with parameter x
def my_function(x):
  return x + 1

# invoke our function
print(my_function(2))
# outputs: 3
print(my_function(3 + 5))
# outputs: 9
```

Functions: Test your knowledge

1: Create a function called `myAdd`

2: Give two parameters to your function

3: Return the sum of the two parameters

4: Test your function

Bonus: Create a function to multiply numbers

A quick note on Modules

Many different modules within python that can be accessed by using import

```
# Three different ways to import modules:
# First way
import module
module.function()

# Second way
from module import function
function()

# Third way
from module import *
function()
```

Can also import the content of another file in the same way and use `as` to assign a module an alias

YOU DID IT!

In this session we have covered basic python coding principles!

You can code!

Now for some challenges...

Before we get started….

In this Intro to Python we have NOT covered:

Classes!

Dictionaries!

Strings!

Visit codecademy for a comprehensive intro: https://www.codecademy.com/learn/learn-python-3

Challenge

CodeWars - Convert boolean values to strings 'Yes' or 'No'.

Complete the method that takes a boolean value and return a "Yes" string for true, or a "No" string for false.

Challenge

CodeWars: Century From Year

Introduction

The first century spans from the **year 1** *up to* and **including the year 100**, **The second** - *from the year 101 up to and including the year 200*, etc.

Task :

Given a year, return the century it is in.

Input , Output Examples ::

```
centuryFromYear(1705)  returns (18)
centuryFromYear(1900)  returns (19)
centuryFromYear(1601)  returns (17)
centuryFromYear(2000)  returns (20)
```

Hope you enjoy it .. Awaiting for Best Practice Codes

Enjoy Learning !!!

Learning Resources

DataQuest https://www.dataquest.io

W3 Schools https://www.w3schools.com/python/

FreeCodeCamp https://guide.freecodecamp.org/python/

Software Carpentry https://software-carpentry.org/

Codecademy https://www.codecademy.com/learn/learn-python-3

Articles and tutorials on Medium, dev.to

Local groups & events

InnovateHer - Get your head around code

Tues 17th Sept, The Plaza

HiPy

Weds 25th Sept, CTL Liverpool

Ladies of Code Liverpool

2nd Wednesday of every month, DoES Liverpool

CodeUp Liverpool

Monthly meetup, Liverpool Science Park

Liverpool Software Developers Meetup

Weds 16th Oct, The Cotton Exchange Building