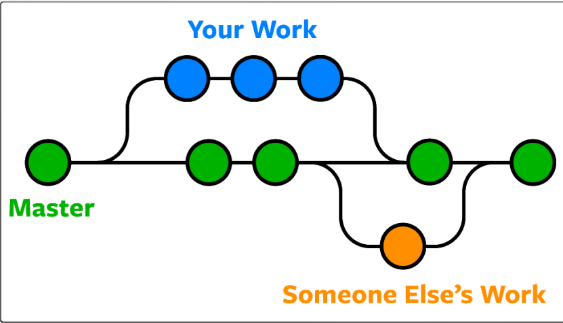
**Weihnachtsprüfung Herr Noel**

**Github:**

**Feature Branch:**

Wird benutzt, um:

* Anzahl der Merge und Konflike zu reduzieren
* Ungestörte Entwicklung der Features
* Commits für ein Feature einfach finden
* Teillieferung von Features
* Änderungen aus dem Master Branch können geschützt werden

Ablauf:

Masterbranch:

1. Git checkout master
2. Git pull --ff-only

Feature Branch anlegen und bearbeiten:

1. Git checkout -b featureeins
2. Git add \*
3. Git commit -m „Beschreibung“

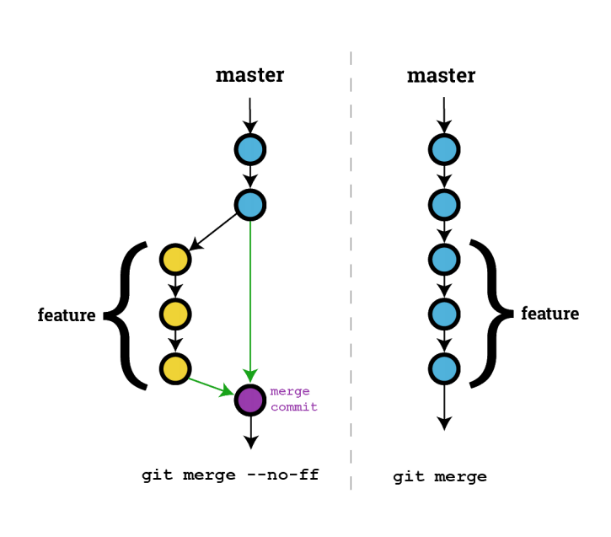
Feature Branch im Remote Repository spreichern:

1. Git push -set-upstream origin featureeins 🡨 nur beim 1. Mal nötig

Feature Branch in Master Branch übertragen:

1. PullRequest anlegen
2. Merge geschiet über den Pull request

Master Branch in den Feature Branch übernehmen:

1. Git checkout master
2. Git pull --ff-only
3. Git checkout featureeins
4. Git merge --no-ff master 🡨 ff = Fast-Forward und wird benutzt, um zu vermeiden, dass der Feature Branch und der Master Bran fusionieren. Außerdem bleiben Commits vom Feature Branch im Feature Branch und es entsteht ein zusätzliches Commit

**Speicherung der Logindaten:**

* Git config credential.helper manager
* Git config user.name bib-weber-darius

**Automatische Wahl des Branches für Pull/Push:**

* Git checkout --track origin/featureeins 🡪 Verbindet einen schon existierenden lokalen Branch mit dem Repository
* Git branch -u origin/featurezwei (auf dem lokalen Branch) 🡪 Erstellen eines neuen lokalen Branchs und diesen mit dem Repository zu verbinden

**Python:**

Global für eine Globale Variabel

Datetime: From Datetime import Datetime. Für Datum 🡪 datetime(1992, 2,28)

Event.wait(5) 🡨 muss 5 warten nach event gesetzt wird

// 🡨 wird zur Ganzzahl Division verwendet

**Just Funktion:**

S = „Hallo“

* s.rjust(10) 🡪 “ Hallo“
* s.ljust(10) 🡪 “Hallo “
* s.center(10,“\_“) 🡪“\_\_Hallo\_\_\_“ 🡨 “\_“ ist Platzhalter

**Split Funktion:**

S = “a“,“b“,“c“,“d“

Print(s.split(“/“)) 🡪 [“a“,“b“,“c“,“d“]

**Join Funktion:**

liste = [“a“,“b“,“c“,“d“]

print(“/“).join(liste) 🡪 a/b/c/d

**Testen ob etwas in einer Liste vorhanden ist:**language = [“de“,“fr“,“en“]  
“ne“ **in** language 🡪 False  
“de“ **in** language 🡪 True  
“d“ **in** language 🡪 True

**Dictionarys:**

Erzeugung 🡪 alter = {'Peter':31, 'Julia':28, 'Werner':35}  
Element hinzufügen 🡪 alter['Moritz'] = 22   
Element löschen 🡪 **del** alter ['Julia']  
Element enthalten? 🡪 if 'Julia' **in** alter  
Anzahl Elemente 🡪 **len**(alter)  
Aktualisierung mit zweiten Dictionary 🡺 ualter = {'Moritz':18, "Werner": 29} 🡪 alter.update(ualter)  
Value 🡺 w = alter.values() 🡪 Values sind nach dem Doppelpunkt also 31, 28, 35 in alter  
Keys 🡺 k = alter.keys() 🡪 Keys sind vor dem Doppelpunkt also Peter, Julia, Werner  
items 🡺 i=alter.items() 🡪 Items sind Key und Value zusammen ('Peter', 41), ('Julia', 28), ('Franz', 35)  
Dictionary umkehren 🡺 alter\_umgekehrt = {}

for name in alter:

personenalter = alter[name]

alter\_umgekehrt[personenalter] =name 🡨 wird zugewiesen

**Formatierungen:**

**Bei Strings:**

“{0:x>10.5s}“.format(“Büllingen“) 🡪 xxxxxBülli

x = Füllzeichen > = rechtsbündig 10=gesamtlänge der Strings 5=String wird auf 5. Stelle gekürzt s=Format String  
“{0:>8.3}|{0:<8.3}|{0:^8.3}“.format(“Peter“) 🡪 “ Pet|Pet | Pet “

**Bei Zahlen:**

“{:=+010.5f}“.format(50.987654321) 🡪 + 050.98765

Beim “=+“ = wird immer Vorzeichen angezeigt ob + oder Minus und wegen dem “=“ linksbündig gesetzt| 010 = die erste 0 ist das Füllzeichen (geht nur bei 0) und die 10 ist die Länge der Zahl mit Vorzeichen und punkt. Die .5f gibt an wie viele Nachkommastellen die Zahl haben soll und dass es ein float ist. Für Dezimal anzugeben verwendet man “d“. Für binär “b“ und Hexadezimal “x“. Für Wissenschaftliche Notation “e“.

**Bei Daten:**

Datum = datetime.now()  
“{:%d/%m/%Y %M:%H}“.format(datum) 🡪 02/49/2020 49:18  
d=tag, m=Monat, Y=Jahr, M=Minute, H=Stunde  
"{:%A}".format(Datum) 🡪 Wednesday gibt also den kompletten Wochentags Namen in lokaler Sprache an und %a die abgekürzte Version.

**Andere Funktionen:**

Variablen: "{vorname} {nachname} {alter}".format(vorname="Hans", nachname="Peter",alter="18")   
🡪 Hans Peter 18

**Mit Dictionarys:** daten = {"vorname":"Hans","nachname":"Peter","alter":18}  
🡪 "{vorname} {nachname} {alter}".format(\*\*daten) 🡪 **die “\*\*“ müssen bei Dictionarys angegeben werden!**